(2)

RL-TR-91-284
Final Technical Report
November 1991

ROME LABORATORY

PERMUTATION SETS AND ROUTABILITY OF MULTISTAGE INTERCONNECTION NETWORKS (MINs)

Syracuse University

Sponsored by
Strategic Defense Initiative Office

APPROVED FOR PUBLIC RELEASE; DISTRIBUTION UNLIMITED.

Rome Laboratory
Air Force Systems Command
Griffiss Air Force Base, NY 13441-570

RL-TR-91-284 has been reviewed and is approved for publication.

APPROVED: Jon B Valente

JON B. VALENTE
Project Engineer

FOR THE COMMANDER: [illegible]

RAYMOND P. URTZ, JR.
Technical Director
Command & Control Directorate

PERMUTATION SETS AND ROUTABILITY OF MULTISTAGE INTERCONNECTION NETWORKS (MINs)

C.Y. Roger Chen
Calvin J.A. Hsia

Contractor: Syracuse University
Contract Number: F30602-88-D-0027
Effective Date of Contract: September 1988
Contract Expiration Date: September 1991
Short Title of Work: Permutation Sets and Routability of MINs
Period of Work Covered: Jul 89 - Sep 89

Principal Investigator: Dr. C.Y. Roger Chen
Phone: (315) 443-4301

RL Project Engineer: Jon B. Valente
Phone: (315) 330-3241

This research was supported by the Strategic Defense Initiative Office of the Department of Defense and was monitored by Jon B. Valente, RL (C3AB) Griffiss AFB NY 13441-5700 under Contract F30602-88-D-0027.

REPORT DOCUMENTATION PAGE

Form Approved
OMB No. 0704-0188

1. AGENCY USE ONLY (Leave Blank)	2. REPORT DATE	3. REPORT TYPE AND DATES COVERED
	November 1991	Final Jul 89 - Sep 89

4. TITLE AND SUBTITLE
PERMUTATION SETS AND ROUTABILITY OF MULTISTAGE INTERCONNECTION NETWORKS (MINs)

5. FUNDING NUMBERS
C - F30602-88-D-0027
Task B-9-3633
PE - 63223C
PR - 2300 (Prev B413)
TA - 03
WU - PP

6. AUTHOR(S)
C.Y. Roger Chen, Calvin J.A. Hsia

7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES)
Syracuse University
Dept of Electrical & Computer Engineering
Syracuse NY 13244-1240

8. PERFORMING ORGANIZATION REPORT NUMBER
N/A

9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES)
Strategic Defense Initiative Office, Office of the Secretary of Defense
Wash DC 20301-7100

Rome Laboratory (C3AB)
Griffiss AFB NY 13441-5700

10. SPONSORING/MONITORING AGENCY REPORT NUMBER
RL-TR-91-284

11. SUPPLEMENTARY NOTES
Rome Laboratory Project Engineer: Jon B. Valente/C3AB/(315) 330-3241

12a. DISTRIBUTION/AVAILABILITY STATEMENT
Approved for public release; distribution unlimited.

12b. DISTRIBUTION CODE

13. ABSTRACT (Maximum 200 words)
Multistage interconnection network is of the most important components for designing high-performance parallel supercomputers and for providing a powerful reconfigurable programming environment such that programming is independent of the actual computer architectures. In this research, some fundamental issues in multistage interconnection networks are investigated. Results, which promise great potential for the design of a reconfigurable high-speed parallel supercomputer with an architecture independent programming environment, are presented. First, two characteristic functions are introduced to characterize networks in a proposed class of multistage interconnection networks. Message routing schemes, network partitioning algorithms, and many other useful properties are presented. Next, a more general class of multistage interconnection networks and two more general characteristic functions are introduced. The transformation rules for one network to emulate another in the class is presented such that the programs and algorithms developed on one machine can be reused on others. Then, the permutation capability (in terms of non-conflict parallel communication) of each network is presented through the concept of network partitioning. An algorithm to determine the admissibility of any permutation on a multistage interconnection network is developed. Finally, a fault-tolerant reconfiguration scheme is presented for parallel processor systems which employ multistage interconnection networks for interprocessor communication. (See reverse)

14. SUBJECT TERMS
Computer Networks, Computer Architectures, Fault Tolerant Computing

15. NUMBER OF PAGES
196

16. PRICE CODE

17. SECURITY CLASSIFICATION OF REPORT	18. SECURITY CLASSIFICATION OF THIS PAGE	19. SECURITY CLASSIFICATION OF ABSTRACT	20. LIMITATION OF ABSTRACT
UNCLASSIFIED	UNCLASSIFIED	UNCLASSIFIED	UL

Block 13 (Cont'd)

Since faults in networks will destroy the communication paths between some network input and output ports, multiple passes through the network are required for the communication between some processor pairs. A shortest-path message routing scheme is developed on such a system. Moreover, fault-tolerant partitioning algorithms are developed such that in each subsystem, the property of full connectivity is maintained.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

CHAPTER 1

INTRODUCTION

The basic concept in designing parallel supercomputers is to employ an interconnection to interconnect a large number of processors and memory modules such that memory modules can be accessed in parallel processors can communicate with each other with serious communication conflict or traffic congestion. Since the trend in designing supercomputers is to use off-the-shelf products for processors and memory modules, the bottleneck is the interconnection network. Specifically, we define an interconnection network to be a connection of switches and links that allows data communication between processors and/or memory modules in a system consisting of multiple processors. Many factors are involved in determining the cost-effectiveness of a particular network design, including the computational tasks it will be used for, the desired speed of interprocessor data transfers, the actual hardware implementation of the network, the number of processors in the system, and cost constraints on the construction [Bhu87]. Interconnection networks can be classified into two categories based on network topologies: *static networks* and *dynamic networks* [Hwa87]. Examples of static networks are linear array, ring, star, tree, mesh, systolic array, chordal ring, hypercube, and cube-connected-cycle. Examples of dynamic networks are single-stage, crossbar, and multistage interconnection networks. The two major switching methodologies ɪor interconnection networks are *circuit switching* and *packet switching*. In circuit switching, a physical path is actually established between a source and a destination. In packet switching, data packets are routed through the interconnection network without establishing a physical connection path. In general, circuit switching is much more suitable for bulk data transmission, and packet switching is more efficient for short data messages.

In this report, we will focus on the domain of multistage interconnection networks since they offer a flexible environment to meet real-time processing requirements for either multiprocessor or share-memory systems. The functionalities, topological relationship, and fault tolerance of various multistage interconnection networks are discussed in this report. The rest of this report is organized as the following five chapters.

In Chapter 2, two characteristic functions **O** and **I** are introduced to uniquely describe any Multistage Interconnection Networks (MINs) constructed by general shuffle connections. All the MINs constructed by general shuffle connections are shown to be in a class of equivalent Banyan type MINs, named as the $\log_2 N$-stage Column-Permute interconnection network ($\log_2 N$ CP^{min}). Based on these two characteristic functions, we show that routing algorithms, network construction rules, network equivalence properties, and network transformation rules can be directly established. As the design of reconfigurable systems is considered, we show that the equivalence among networks can easily be described by linear transformations on characteristic functions. In other words, the equivalence can be interpreted as a renaming scheme on the inputs and outputs of a network. We explain why the routing scheme on each network is always destination-oriented and source-preserved.

In Chapter 3, the CP^{min} networks in Chapter 2 are extented to a class of Bit-Permute-Complement (BPC) type multistage interconnection networks. This class of BPC networks is based on a more general interconnection model than that of CP^{min}. Typically, they possess a very simple routing scheme such that for any communication path the source address can be easily preserved and the destination can be used as routing tags. General rules for transforming a multistage interconnection network into another in this class by simply renumbering the inputs and outputs of the network are presented. Both distributed and global routing schemes after the transformation are discussed. By using the proposed network transformation rules, algorithms developed on a machine using a multistage interconnection network can be directly used on another machine which employs a different network.

In Chapter 4, the permutation capability of the class of BPC multistage interconnection networks defined by a general model using bit-permute-complement connections, which includes the Omega network, baseline network, Indirect binary cube network, etc. as special cases, is studied. In this chapter, several questions are addressed. How can we easily characterize all the admissible permutations of a network? How can we determine whether or not a permutation is admissible on a network? We start our discussion on Omega networks due to their regular structure, and then generalize the problem to the general model. We show that the set of admissible permutations of a network can be characterized by very simple bit relations depending on two characteristic functions which specify this network. The time complexity of our proposed algorithm to determine the admissibility of a permutation on a multistage interconnection network is $O(N)$, where N is the number of inputs/outputs of the network.

In Chapter 5, the fault tolerance capability of a multistage interconnection network and the technique to reconfigure a network under multiple faults are discussed. Both faulty switching elements and faulty communication links are considered in our fault model. Generally speaking, in a multistage interconnection network used for interprocessor connections, if faults occur, many input-output communication paths will no longer be available. A solution to this is to allow data propagation through the faulty network multiple passes, such that reasonable communication capability can be maintained. In this chapter, a fault-tolerant reconfiguration scheme is developed for an N-processor system interconnected by a $\log_2 N$-stage Omega network under multiple faults. Regardless of whether the faults on the Omega network are critical or not, a deadlock-free environment is provided for the N-processor system by applying our reconfiguration scheme. The reconfiguration of such a multiprocessor system is based on three principles: disable processors whose communication capabilities are completely destroyed, eliminate faulty components and, if necessary, sacrifice some usable components implicitly without knowing the actual locations of these components. The reconfigured system is a surviving system such that it may be an intergrated one consisting of a subset of the N processors (including the case of all the N processors) or it may be partitioned into a number of subsystems such that the dynamic full access property within each subsystem (system) can be maintained. A deadlock-free shortest-path routing table is obtained for each processor in the surviving system (subsystem) to avoid the danger of deadlock traps caused by uncautiously using unidirectional communication paths rather than bidirectional communication paths between some processors. The time complexity of our reconfiguration scheme is analyzed as well.

CHAPTER 2

CHARACTERIZATION OF MULTISTAGE INTERCONNECTION NETWORKS

2.1. INTRODUCTION

The use of Multistage Interconnection Networks (MINs) is considered a very cost-effective means to provide efficient interprocessor and/or processor-memory communication [Fen81] [WuFe80] [Sie79]. Examples are the Omega network [Law75], baseline and reverse baseline network [WuFe80], indirect binary cube network [Pea77], Delta network [Pat81], cube network [SiSm78], flip network [Bat76] and modified data manipulator [Fen74]. It is well known that all those MINs are constructed through a particular type of connections called *general shuffle connections*. It is clear that, in addition to those seven MINs, there exists a huge set of MINs which are also constructed by general shuffle connections. It is also well known that the capability of each MIN in terms of non-conflicting communication permutations is different. For example, it has been reported in [NaSa81] that an Omega network can realize cyclic shifts, p-ordering, inverse p-ordering, etc. and cannot realize operations such as bit-reversal. Since each application will require a different set of permutations in order to optimally perform the execution, it is extremely important for a system designer to be able to identify a MIN which is best suited for applications needs. However, due to the lack of a general understanding of network characteristics, so far, a designer has to select a MIN out of the seven known MINs (instead of out of millions of MINs) in a rather *ad hoc* manner. This has greatly limited the achievable performance of a parallel supercomputer. Since it is impossible to investigate the millions of MINs one by one, it is essential to be able to characterize those MINs such that a precise quantitative description of them will then be available and a MIN can be immediately investigated as soon as the characteristic functions are available. Since the characteristic functions of this MIN will carry information regarding the permutation capability of this MIN, it will then be possible for a designer to select a MIN out of millions of MINs to optimally meet application needs. In this chapter, we will show how to characterize the whole set of MINs which are constructed by general shuffle connections.

Many theoretical properties of the above seven MINs have been discussed, which are summized as follows. Their inputs and outputs are fully connected and a simple routing scheme can be applied from any input to any output [Fen81] [WuFe80] [Sie79] [Law75] [Pea77] [Pat81] [SiSm78] [Bat76] [Fen74]. They have the *buddy property* [WuFe80] [Agr83] [AgSw88] and belong to *bidelta networks* [KrSn86]. The topological equivalence problem among them has been exhibited based on the fact that their graph representations are isomorphic to the Banyan graph [GoLi78]; that is, they are all equivalent Banyan type MINs. In [BeFo88], Bermond and Fourneau discussed some Banyan type MINs with *independent connections* and showed an approach to explain their topological equivalence. In [KrSn86], Huang and Tripathi have shown that a MIN can be represented by a finite state machine. However, other than issues of theoretical interest, these works still tell us little about how to help designers to investigate or understand the capabilities of the other millions of MINs. Therefore, their contribution towards the practical application needs which we mentioned earlier are rather limited. This again shows that, in addition to the discussion of common topological properties, it is extremely important to be able to characterize MINs and directly discuss the network capability, routing algorithms, and construction rules on the characteristic functions. We believe that this is the starting point for system designers to fully exploit the capabilities of MINs and really take maximal advantage of them.

In this chapter, our concern is to give a global view of the characteristics of various MINs. First of all, the topological structure of the whole class of $\log_2 N$-stage Banyan type MINs constructed by general shuffle connections (which are called *column permute connections* in this chapter) is defined in detail, which includes all the seven mentioned MINs as special cases. We denote them as $\log_2 N$ CP^{min}. A *path-descriptive* methodology is used to characterize topological features on the whole class and we show that two important permutation functions, **O** and **I**, can uniquely specify each MIN in the class. We call these two *characteristic functions*. Based on functions **O** and **I**, many important features of MINs including routing algorithms, network construction, permutation capability, etc. can be immediately established or examined. As the topological equivalence is considered, we show that the equivalent relation among MINs can be subdivided and is an intrinsically linear transformation on functions **O** and **I**. In other words, the transfer from one MIN to another can be viewed as a renaming scheme on the inputs and outputs of a MIN. It has a significant impact on the design of a reconfigurable system. As communication is considered, we show that the routing scheme for each MIN in the class can be directly derived from the characteristic functions **O** and **I**. We explain why the distributed routing algorithm for each MIN is *destination-oriented* (i.e., its routing tags can be determined by the destination addresses only) and *source-preserved* (i.e., the address of the source input can automatically be preserved without extra efforts). Note that, in this chapter, since we limit our discussion to $\log_2 N$-stage MINs only, $\log_2 N$ $\mathrm{CP}^{\min}$ and $\mathrm{CP}^{\min}$ are used interchangeably.

The rest of this chapter is organized as follows. In Section 2.2, we introduce a class of basic connection patterns defined by CP(n) permutations. Section 2.3 is devoted to the topological structure of the $\mathrm{CP}^{\min}$ and the outline of two important characteristic functions, **O** and **I**. We show how they are related to the equivalence and transformation among MINs. In Section 2.4, we show the relation between general routing algorithms and the characteristic functions. In Section 2.5, the decomposition and partitioning of various networks are discussed. Finally, Section 2.6 concludes this chapter.

2.2. BASIC CONNECTION PATTERN

In order to describe a network in terms of permutations on N symbols, we may label the links of this network at the inputs and outputs of all switching elements following their natural order in the drawing. A label is a number between 0 and N-1 whose binary representation (i.e., address) can be denoted by $(x_{n-1}, \ldots, x_1, x_0)$, where x_0 is the least significant bit (LSB). Each connection link is defined by two labels and each connection pattern between two adjacent switching stages is specified by a permutation of N labels.

An n-stage MIN constructed with $n+1$ connection patterns has often been defined using these premutations [Par80]. For example, the Omega network is defined as n consecutive perfect shuffle permutations plus an identity permutation for connecting outputs of the network. A perfect shuffle permutation is defined as a circular left shift of the binary representation of an operand.

Note that not every arbitrarily selected $n+1$ connection patterns can be used to design a MIN with the desired properties. For example, any disturbance in the order of $n+1$ connection patterns on the Omega network may result in a new network which will no longer preserve the full connectivity property. In other words, the topological structure of a MIN is closely related to its functional behavior and its construction should be based on a systematic method for selecting a sequence of valid connection patterns.

In this section, we introduce a class of basic connection patterns. Using this set of permutations, we can define construction rules and describe the functional behavior of various MINs.

Consider numbers from 0 to N-1 and the binary representation of each of them, $X = (x_{n-1}, \ldots, x_1 x_0)$. We define the class CP(n) of Column-Permute premutation functions by a permutation on indices of the representation.

DEFINITION 1: A CP permutation **P** in CP(n) is specified by an n-tuple vector $V = (\theta(n-1), \ldots, \theta(1), \theta(0))$, where θ is a permutation function on $(n\text{-}1, \ldots, 1, 0)$ and $(\theta(n-1), \ldots, \theta(1), \theta(0))$ is the image of θ. The mapping of P on $X = (x_{n-1}, \ldots, x_1 x_0) \in [0, N\text{-}1]$ is obtained as follows:

$$\mathbf{P} \in \mathrm{CP}(n) \quad (N = 2^n)$$

$$\begin{aligned}\mathbf{P}(x_{n-1}, \ldots, x_1 x_0) &= (\mathbf{P}(x_{n-1}), \ldots, \mathbf{P}(x_1), \mathbf{P}(x_0)) \\ &= (x_{\theta(n-1)}, \ldots, x_{\theta(1)}, x_{\theta(0)}) \\ &= (y_{n-1}, \ldots, y_1 y_0). \qquad \square\end{aligned}$$

It is very easy to show that the **P** operation is closed with respect to the domain $[0, N\text{-}1]$, i.e., $\mathbf{P}(X) \in [0, N\text{-}1]$ for all $X \in [0, N\text{-}1]$. Similarly, we can define the inverse function $\mathbf{P}^{-1}$ of

P.

DEFINITION 2: Let θ^{-1} be the inverse permutation function of θ. The inverse CP permutation $\mathbf{P}^{-1}$ of $\mathbf{P}$ defined in Definition 1 is specified by an n-tuple vector V^{-1} = $(\theta^{-1}(n-1), \ldots, \theta^{-1}(1), \theta^{-1}(0))$. The mapping of $\mathbf{P}^{-1}$ on $X = (x_{n-1}, \ldots, x_1, x_0) \in [0, N-1]$ is obtained as follows:

$$\mathbf{P}^{-1} \in \mathrm{CP}(n) \quad (N = 2^n)$$

$$\begin{aligned}\mathbf{P}^{-1}(x_{n-1}, \ldots, x_1, x_0) &= (\mathbf{P}^{-1}(x_{n-1}), \ldots, \mathbf{P}^{-1}(x_{-1}), \mathbf{P}^{-1}(x_0)) \\ &= (x_{\theta^{-1}(n-1)}, \ldots, x_{\theta^{-1}(1)}, x_{\theta^{-1}(0)}) \\ &= (z_{n-1}, \ldots, z_1, z_0).\end{aligned}$$ □

For example, consider $n = 4$ and let $V = (2,1,0,3)$ and $X = (1,0,0,1)$. We have $y_3 = x_2$, $y_2 = x_1$, $y_1 = x_0$ and $y_0 = x_3$. Hence, $\mathbf{P}(X) = (0,0,1,1)$. Similarly, we have $V^{-1} = (0,3,2,1)$ and $\mathbf{P}^{-1}(X) = (1,1,0,0)$.

Note that the k-subshuffle σ_k, the perfect shuffle σ (i.e., the $(n$-1)-subshuffle), the k-butterfly β_k and the bit reversal ρ are defined as follows:

$$\sigma_k(X) = (x_{n-1}, \ldots x_{k+1}, x_{k-1}, \ldots x_0, x_k)$$

$$\sigma(X) = (x_{n-2}, \ldots x_1, x_0, x_{n-1})$$

$$\beta_k(X) = (x_{n-1}, \ldots x_{k+1}, x_0, x_{k-1}, \ldots x_1, x_k)$$

$$\rho(X) = (x_0, x_1, \ldots x_{n-2}, x_{n-1})$$

These functions are examples of general shuffle permutations and are used to define the basic connection patterns to design the six MINs studied by Wu and Feng [WuFe80]. It is particularly worthwhile to discuss a special subset of permutations in CP(n) with $\theta(0) = 0$, because these permutations give remarkable restrictions on the design of useful n-stage MINs as we shall see later.

As the connection patterns defined by N-symbol permutations are considered, the switching elements on stages at both sides of a connection pattern can be labeled from 0 to $\frac{N}{2}$ -1 = 2^{n-1}-1 following the same ordering as the labeling of their output (input) links. That is, let Y = $(x_{n-1}, \ldots, x_1)$ be the label of a switching element, then the output (input) links connected to this switching element are labeled

$$X_0 = (x_{n-1}, \ldots, x_1, 0)$$

$$X_1 = (x_{n-1}, \ldots, x_1, 1).$$

These X_0 and X_1 are named as the 0-output (0-input) link and 1-output (1-input) link with respect to Y, respectively. For a connection pattern defined by a CP permutation **P** connecting two adjacent switching stages, say stage j and stage $j+1$ (excluding the special case specified by $\theta(0) = 0$), the 0-output link $(x_{n-1}, \ldots, x_1, 0)$ of a switching element Y = $(x_{n-1}, \ldots, x_1)$ at stage j is connected to the input link $(\mathbf{P}(x_{n-1}), \ldots, \mathbf{P}(x_1), \mathbf{P}(x_0))$ of a switching element $(\mathbf{P}(x_{n-1}), \ldots, \mathbf{P}(x_1))$ at stage $j+1$ such that some $\mathbf{P}(x_i) = 0$, $1 \le i \le n-1$. Similarly, the 1-output link $(x_{n-1}, \ldots, x_1, 1)$ is connected to the input link $(\mathbf{P}(x_{n-1}), \ldots, \mathbf{P}(x_1), \mathbf{P}(x_0))$ of a switching element $(\mathbf{P}(x_{n-1}), \ldots, \mathbf{P}(x_1))$ such that $\mathbf{P}(x_i) = 1$. These two switching elements which connect to the 0-output link and 1-output link of Y are called Y's 0-successor and 1-successor, respectively. We denote them as $succ^0(Y)$ and $succ^1(Y)$. In a similar way, for a switching element $Z = (z_{n-1}, \ldots, z_1)$ at stage $j+1$, $prec^0(Z)$ and $prec^1(Z)$ at stage j which connect the 0-input link and 1-input link of Z represent Z's 0-predecessor and 1-predecessor, respectively. As those connection patterns defined by CP permutations with $\theta(0) = 0$ are considered, we have $succ^0(Y) = succ^1(Y)$ and $prec^0(Z) = prec^1(Z)$. Therefore, for a connection pattern defined by a $\mathbf{P} \in \mathrm{CP}(n)$ connecting stage j and stage $j+1$, we can always have the following relation: let $Y = (u_{n-1},...,u_1)$, $Y \in [0, \frac{N}{2} -1]$, be a switching element at stage j. If $\mathbf{P}$ is specified by $\theta(0) \neq 0$, then $succ^1(Y) - succ^0(Y) = 2^i$ for some $i \in [1, n-1]$; else $succ^1(Y) - succ^0(Y) = 0$. Similarly, let $Z = (v_{n-1}, \ldots, v_1)$, $Z \in [0, \frac{N}{2} -1]$, be a switching element at stage $j+1$. If $\mathbf{P}$ is specified by $\theta(0) \neq 0$, then $prec^1(Z) - prec^0(Z) = 2^k$ for some $k \in [1, n-1]$; else $prec^1(Z) - prec^0(Z) = 0$.

Our CP(n) connection patterns satisfy the definition of independent connections [BeFo88]. Thus, they are independent connections, too. For example, in Fig. 2.1, we show a connection pattern defined by a CP(4) permutation on numbers from 0 to 15. The permutation is specified by $V = (0,1,2,3)$ which is a bit reversal function ρ. In Fig. 2.2, we show another connection pattern specified by the vector $V = (1,3,2,0)$ in which $\theta(0) = 0$.

2.3. TOPOLOGICAL STRUCTURE OF THE $\log_2 N$ CP$^{\mathrm{min}}$

A MIN is said to have the *Banyan Property* if and only if for any input and output there exists a unique path connecting them, i.e., its inputs and outputs are fully connected. Any Banyan type MIN can easily be modeled by a Banyan graph in which vertices represent switching elements and arcs represent connection links [Agr83]. The structure of a Banyan graph is essentially an overlay of tree structures and assures full connectivity among base and apex vertices without redundancy. In particular, those $N \times N$ n-stage Banyan type MINs [Fen81] constructed with 2×2 switching elements can be modeled by $(2,2,n)$ rectangular SW Banyan graphs and its corresponding Banyan graph representation. As mentioned in some

$$\begin{array}{cc}
X & \varrho(X) \\
\begin{bmatrix}
0 & 0 & 0 & 0 \\
0 & 0 & 0 & 1 \\
0 & 0 & 1 & 0 \\
0 & 0 & 1 & 1 \\
0 & 1 & 0 & 0 \\
0 & 1 & 0 & 1 \\
0 & 1 & 1 & 0 \\
0 & 1 & 1 & 1 \\
1 & 0 & 0 & 0 \\
1 & 0 & 0 & 1 \\
1 & 0 & 1 & 0 \\
1 & 0 & 1 & 1 \\
1 & 1 & 0 & 0 \\
1 & 1 & 0 & 1 \\
1 & 1 & 1 & 0 \\
1 & 1 & 1 & 1
\end{bmatrix}
&
\begin{bmatrix}
0 & 0 & 0 & 0 \\
1 & 0 & 0 & 0 \\
0 & 1 & 0 & 0 \\
1 & 1 & 0 & 0 \\
0 & 0 & 1 & 0 \\
1 & 0 & 1 & 0 \\
0 & 1 & 1 & 0 \\
1 & 1 & 1 & 0 \\
0 & 0 & 0 & 1 \\
1 & 0 & 0 & 1 \\
0 & 1 & 0 & 1 \\
1 & 1 & 0 & 1 \\
0 & 0 & 1 & 1 \\
1 & 0 & 1 & 1 \\
0 & 1 & 1 & 1 \\
1 & 1 & 1 & 1
\end{bmatrix}
\end{array}$$

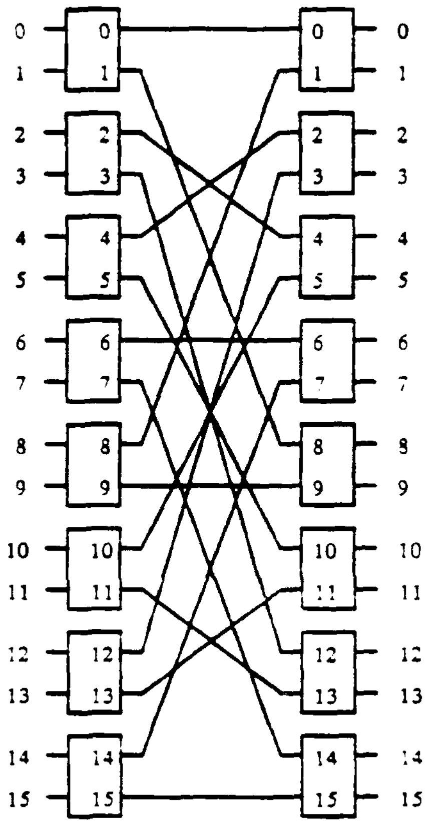


Fig. 2.1. The connection pattern defined by a CP(4) permutation $V = (0,1,2,3)$, i.e., a bit reversal function ϱ .

$$X = \begin{bmatrix} 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 1 & 1 \\ 0 & 1 & 0 & 0 \\ 0 & 1 & 0 & 1 \\ 0 & 1 & 1 & 0 \\ 0 & 1 & 1 & 1 \\ 1 & 0 & 0 & 0 \\ 1 & 0 & 0 & 1 \\ 1 & 0 & 1 & 0 \\ 1 & 0 & 1 & 1 \\ 1 & 1 & 0 & 0 \\ 1 & 1 & 0 & 1 \\ 1 & 1 & 1 & 0 \\ 1 & 1 & 1 & 1 \end{bmatrix} \qquad \mathbf{P}(X) = \begin{bmatrix} 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 1 & 0 & 0 & 0 \\ 1 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 1 & 1 \\ 1 & 0 & 1 & 0 \\ 1 & 0 & 1 & 1 \\ 0 & 1 & 0 & 0 \\ 0 & 1 & 0 & 1 \\ 1 & 1 & 0 & 0 \\ 1 & 1 & 0 & 1 \\ 0 & 1 & 1 & 0 \\ 0 & 1 & 1 & 1 \\ 1 & 1 & 1 & 0 \\ 1 & 1 & 1 & 1 \end{bmatrix}$$

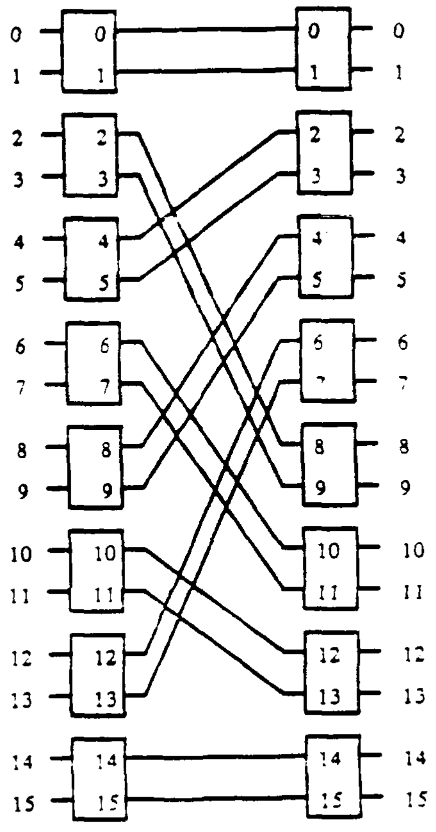


Fig. 2.2. The connection pattern defined by a CP(4) permutation $V = (1,3,2,0)$ in which $\theta(0) = 0$ ·

previous works [WuFe80][Agr83][HuTr86][BeFo88], all the networks with graph representations isomorphic to the same Banyan graph are topologically equivalent.

It is a direct conclusion from Section 2.2 and [BeFo88] that if there exist n-stage Banyan type MINs which can be defined by $(n+1)$-level CP(n) permutations, they must be topologically equivalent to one another and their graph representations are isomorphic to that of a baseline network. What we are concerned with in this section is how to construct a class of Banyan type MINs (i.e., the class of $\mathrm{CP}^{\min}$ in our notation) using connection patterns defined by CP type permutations. In order to give a more intuitive explanation of their functional properties, a *path-descriptive* methodology is adopted hereafter. Our point of view is that for easily routing the message from a source input to a destination output in a MIN, the routing scheme should preserve the information of source and destination addresses, and, what is more significant, indicate the topological structure of this MIN.

Consider an $N \times N$ n-stage MIN, Γ (see Fig. 2.4), consisting of n switching stages and $n+1$ connection patterns defined by CP(n) permutations. Γ can be defined as

$$\Gamma = \mathbf{P}^n \mathbf{E}^{(n-1)} \mathbf{P}^{(n-1)} \cdots \mathbf{E}^0 \mathbf{P}^0$$

where $\mathbf{P}^i \in$ CP(n), $\mathbf{E}^i = \mathbf{E}$ (for all i) denote switching stages, and the superscript i specifies the ith stage. The effect of a switching stage $\mathbf{E}$ is an exchange permutation which is obtained as follows:

For $X = (x_{n-1}, \ldots, x_1, x_0)$, $X \in [0, N-1]$

$\mathrm{E}(X) = (x_{n-1}, \ldots, x_1, d)$

where $d = x_0$ or $\bar{x}_0$ (the complement of x_i).

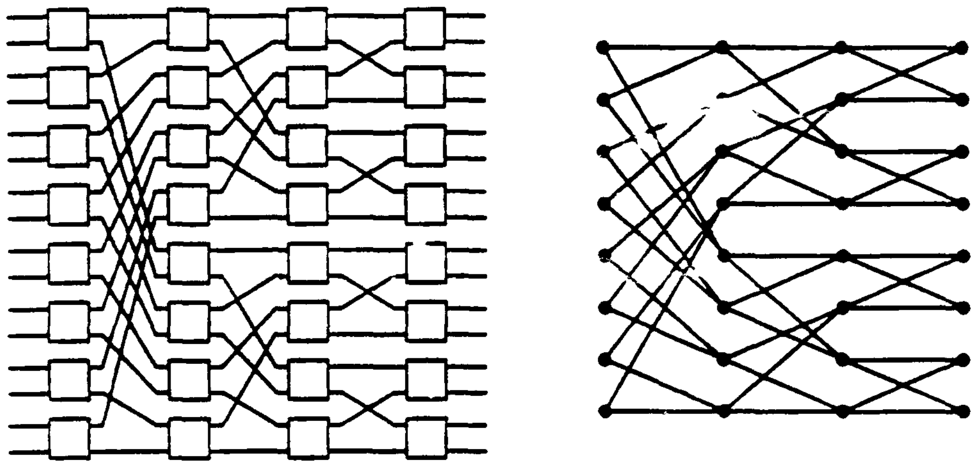


Fig. 2.3. The Baseline network and its Banyan graph representation.

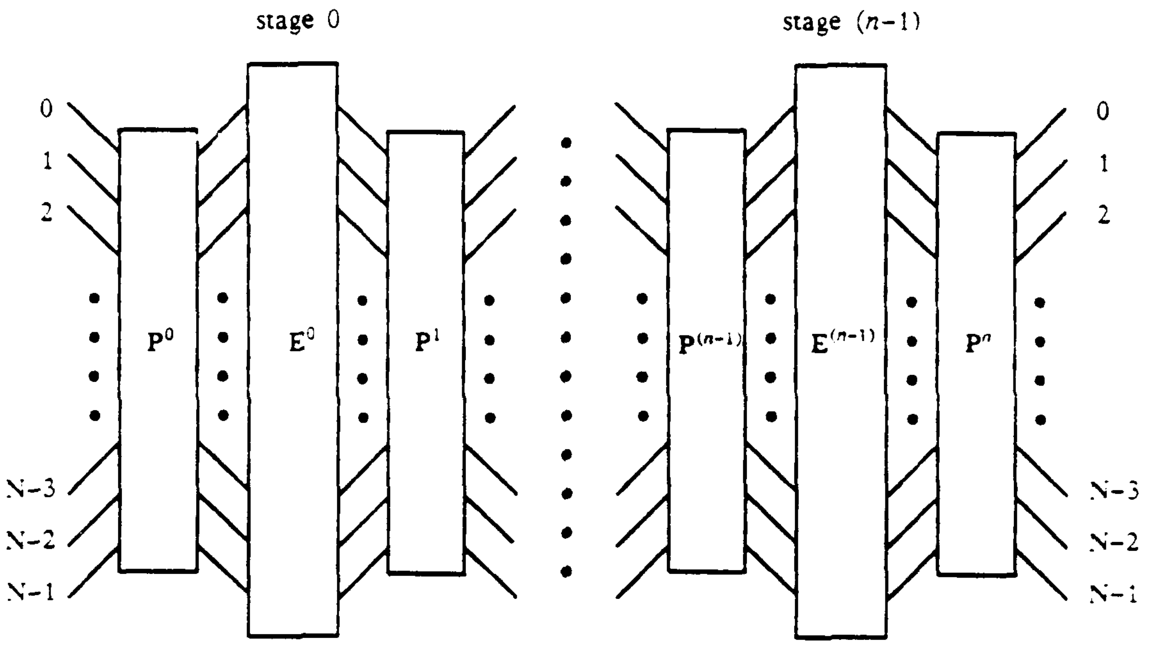


Fig. 2.4. An $N \times N$ n-stage MIN, Γ ,consisting of n switching stages and $(n+1)$-level connection patterns defined by CP(n) permutations.

There are two kinds of operations on Γ. A $\mathbf{P}^i$ permutes binary bits of the operand according to its corresponding vector $V^i = (\theta^i(n-1), \ldots, \theta^i(1), \theta^i(0))$ and an $\mathbf{E}^i$ replaces the least significant bit (LSB) of the operand either by the original bit or its complement. Generally speaking, the result of an operand X performed by $\mathbf{P}^i$ and $\mathbf{E}^i$ consecutively can be expressed as follows:

$$X = (x_{n-1}, \ldots, x_1, x_0)$$

$$X^i = \mathbf{P}^i(X) = (x_{\theta^i(n-1)}, \ldots, x_{\theta^i(1)}, x_{\theta^i(0)})$$

$$= (y^i_{n-1}, \ldots, y^i_1, y^i_0)$$

$$X^{(i)(i)} = \mathbf{E}^i(X^i) = (y^i_{n-1}, \ldots, y^i_1, d^i).$$

Here, we compose functions from right to left so that for operations $\mathbf{P}^i$ and $\mathbf{E}^i$ over $X \in [0, N-1]$, $\mathbf{E}^i\mathbf{P}^i$ is defined as $\mathbf{E}^i(\mathbf{P}^i(X))$.

For Γ to be a Banyan type MIN, there exists a unique path from any input $S = (s_{n-1}, \ldots, s_1, s_0)$ to an arbitrary output $D = (d_{n-1}, \ldots, d_1, d_0)$. Conceptually, we can imagine that S is propagated through Γ to D performed by those $2n+1$ operations consecutively. Hence, we can get a unique valid transition sequence consisting of $2n+1$ binary numbers, i.e.,

$$D^0 = \mathbf{P}^0(S)$$

$$D^{(0)(0)} = \mathbf{E}^0(D^0)$$

$$D^1 = \mathbf{P}^1(D^{00})$$

$$D^{(1)(1)} = \mathbf{E}^1(D^1)$$

$$\ldots\ldots$$

$$D^{(n-1)(n-1)} = \mathbf{E}^{(n-1)}(D^{(n-1)})$$

and eventually,

$$\begin{aligned} D^n &= \mathbf{P}^n(D^{(n-1)(n-1)}) \\ &= D\,. \end{aligned}$$

In their mathematical meaning, D^i and $D^{(i)(i)}$ are mappings of operands $D^{(i-1)(i-1)}$ and D^i performed by $\mathbf{P}^i$ and $\mathbf{E}^i$, respectively. From another point of view, according to the definition of basic connection patterns, we can say that each D^i is the address of the input link through which the path from S to D traverses at stage i. Similarly, each $D^{(i)(i)}$ is the address of the output link traversed stage i. More precisely, the transition sequence has the following form:

$$\begin{aligned} S &= (s_{n-1}, \ldots, s_1, s_0) \\ D^0 &= (s^0_{n-1}, \ldots, s^0_1, s^0_0) \\ D^{(0)(0)} &= (s^0_{n-1}, \ldots, s^0_1, d^0) \\ D^1 &= (s^1_{n-1}, \ldots, s^1_1, s^1_0) \\ D^{(1)(1)} &= (s^1_{n-1}, \ldots, s^1_1, d^1) \\ &\ldots\ldots \\ D^{(n-1)} &= (s^{n-1}_{n-1}, \ldots, s^{n-1}_1, s^{n-1}_0) \\ D^{(n-1)(n-1)} &= (s^{n-1}_{n-1}, \ldots, s^{n-1}_1, d^{n-1}) \\ D^n &= (s^n_{n-1}, \ldots, s^n_1, s^n_0) \\ &= (d_{n-1}, \ldots, d_1, d_0) \\ &= D\,. \end{aligned}$$

Instead of being a description using a graph model, the above is a path-descriptive outline of the structure of an n-stage Banyan type MIN defined by CP(n) permutations. The following theorem is a necessary and sufficient condition for the class of n-stage MINs defined by CP(n) permutations to satisfy the Banyan property.

THEOREM 1: Let Γ be an n-stage MIN defined by CP(n) permutations. Γ is an n-stage Banyan type MIN if and only if there exists an $\mathbf{O} \in$ CP(n) specified by a vector V^O = $(\theta^O(n-1), \ldots, \theta^O(1), \theta^O(0))$ such that

$$\mathbf{O}(s_{n-1}, \ldots, s_1, s_0) = (s_{\theta^O(n-1)}, \ldots, s_{\theta^O(1)}, s_{\theta^O(0)})$$
$$= (s_0^{n-1}, \ldots, s_0^1, s_0^0)$$

is true for each transition sequence representing the path from any input S = $(s_{n-1}, \ldots, s_1, s_0)$ to any output $D = (d_{n-1}, \ldots, d_1, d_0)$. Here, s_0^i is the LSB of D^i in a transition sequence.

PROOF: (only if) Let $i, j, k, l \in [0, n\text{-}1]$. We assume Γ is a Banyan type MIN. There exists a unique path between any input and output of Γ. First, each $\mathbf{P}^i$ in Γ is a fixed permutation pattern which can only permute bits of its operand. Second, in each transition sequence all possible chances to change the value of a bit exist on those n exchange permutations corresponding to n switching stages where LSB's of their operands can be changed. Thus, as in any transition sequence each s_i must be given exactly one chance to be changed to some desired d_j. This is true, because if any s_i gets more than one chance to be changed, then at least one s_l, $l \neq i$, has no chance to be changed, and each input S could reach no more than 2^{n-1} of the total 2^n outputs. This contradicts our assumption. Moreover, for each input S two different transition sequences cannot reach the same output D. They are different in the

LSB of at least one $D^{(i)(i)}$ if and only if the outputs they reach are different. Therefore, each s_i is allowed to appear exactly once at the position of the LSB in some D^k. That is, $s_i = s_0^k$ for some $i, k \in [0, n\text{-}1]$ and there is a one-to-one mapping between the subscript i and superscript k. Obviously, for Γ to be a Banyan type MIN, it is the responsibility of each $\mathbf{P}^k$ where $k \in [0, n\text{-}1]$ to bring some unique s_i in S to the position of the LSB exactly once. Here, s_i gets the only chance to be changed to a desired d_j after the operation of a switching stage. However, $\mathbf{P}^n$ can be an arbitrary permutation in CP(n). Thus, the existence of such a permutation function $\mathbf{O} \in$ CP(n) such that

$$\mathbf{O}(s_{n-1}, \ldots, s_1, s_0) = (s_0^{n-1}, \ldots, s_0^1, s_0^0)$$

is clear. Let function $\mathbf{O}$ be specified by a vector $V^O = (\theta^O(n-1), \ldots, \theta^O(1), \theta^O(0))$. Then V^O represents the order of bits of S to be permuted to the position of LSB in a transition sequence.

(if) Since there exists a function $\mathbf{O} \in$ CP(n) such that

$$\mathbf{O} = (s_{n-1}, \ldots, s_1, s_0) = (s_0^{n-1}, \ldots, s_0^1, s_0^0)$$

is true for all the transition sequences on Γ, each bit of an arbitrary S gets only one chance to appear in the position of LSB where it can be changed to a desired value after the operation of a switching stage. That is to say, any input S has a unique transition sequence or path to reach any one of the 2^n outputs. Therefore, there exists a unique path between any input and any output on Γ. □

In Theorem 1, we outlined one characteristic on the topological structure of the CP^{min} as the necessary and sufficient condition for each MIN in the class to satisfy the Banyan property. For a complete analysis of the CP^{min}, there is still another thing which should be noted on the transition sequence. After each s_i is permuted to the position of the LSB in some D^k, it must be replaced by a desired d_j as the result of $D^{(k)(k)}$, i.e., $d_j = d^k$. Then, in the D^{k+1}, d_0^k is permuted to the position of bit l where $l \neq 0$ (i.e., $d^k = s_l^{k+1}$), because another s_m ($m \neq i$) should appear in the position of LSB as required by the necessary condition that Γ needs to be a Banyan type MIN. As a result, we have the following relation in each D^{i+1}:

$$\{d^i, \ldots, d^0\} \subset \{s_{n-1}^{i+1}, \ldots, s_1^{i+1}\}, \text{ for all } i \in [0, n\text{-}2]$$

and

$$\{d^{n-1}, \ldots, d^0\} = \{s_{n-1}^n, \ldots, s_1^n\}.$$

Eventually, after the mapping of $\mathbf{P}^n$, all the d^k where $k \in [0, n\text{-}1]$ will be arranged in the correct position corresponding to $D = (d_{n-1}, \ldots, d_1, d_0)$ such that $d_k = d_k^n$. Clearly, there exists another permutation function $\mathbf{I} \in \mathrm{CP}(n)$ which is related to the order in which bits of D are to be replaced. Given a $\Gamma \in \mathrm{CP}^{\min}$, the function $\mathbf{I}$ is an inherent characteristic in addition to the function $\mathbf{O}$. We can use $\mathbf{I}$ and $\mathbf{O}$ to uniquely describe the topological structure of a $\mathrm{CP}^{\min}$ type MIN. We call $\mathbf{I}$ and $\mathbf{O}$ *characteristic functions*.

DEFINITION 3: Let Γ be an n-stage $\mathrm{CP}^{\min}$ type MIN. Γ can be characterized by the two CP permutation functions $\mathbf{O}$ and $\mathbf{I}$, named as characteristic functions. The function $\mathbf{O}$ has the same definition as described in Theorem 1. The function $\mathbf{I} \in \mathrm{CP}(n)$ is specified by a vector $V^I = (\theta^I(n-1), \ldots, \theta^I(1), \theta^I(0))$ such that

$$\begin{aligned}(d_{n-1}, \ldots, d_1, d_0) &= \mathbf{I}(d^{n-1}, \ldots, d^1, d^0) \\ &= (d^{\theta^I(n-1)}, \ldots, d^{\theta^I(1)}, \theta^I(0)).\end{aligned}$$

Here, d^i is the LSB of $D^{(i)(i)}$ in a transition sequence. □

Now, we can conclude the above discussion as follows. As long as all the $n+1$ selected connection patterns used to construct a CP^{min} type MIN Γ satisfy Theorem 1, the topological structure of Γ can be specified by two characteristic functions O and I as defined in Definition 3. In other words, Theorem 1 and Definition 3 imply that any transition sequence of a CP^{min} type MIN has the following form:

$$S = (s_{n-1}, \ldots, s_1, s_0)$$

$$D^0 = (s_{n-1}^0, \ldots, s_1^0, s_{\theta^O(0)})$$

$$D^{(0)(0)} = (s_{n-1}^0, \ldots, s_1^0, d^0)$$

$$D^1 = (s_{n-1}^1, \ldots, s_1^1, s_{\theta^O(1)})$$

$$D^{(1)(1)} = (s_{n-1}^1, \ldots, s_1^1, d^1)$$

$$\ldots\ldots$$

$$D^{(n-1)} = (s_{n-1}^{n-1}, \ldots, s_1^{n-1}, s_{\theta^O(n-1)})$$

$$D^{(n-1)(n-1)} = (s_{n-1}^{n-1}, \ldots, s_1^{n-1}, d^{n-1})$$

$$D^n = (s_{n-1}^n, \ldots, s_1^n, s_0^n)$$

$$= (d^{\theta^I(n-1)}, \ldots, d^{\theta^I(1)}, d^{\theta^I(0)})$$

$$= (d_{n-1}, \ldots, d_1, d_0)$$

where $\{d^i, \ldots, d^0\} \subset \{s_{n-1}^{i+1}, \ldots, s_1^{i+1}\}$, for all $i \in [0, n-2]$, and $d^j \in \{d^{\theta^I(n-1)}, \ldots, d^{\theta^I(1)}, d^{\theta^I(0)}\}$, for all $j \in [0, n-1]$.

Thus, the values of $\theta^O(i)$ and $\theta^I(i)$ of a CP^{min} type MIN can be easily obtained from its transition sequence. That is to say, $(\theta^O(n-1), \ldots, \theta^O(1), \theta^O(0))$ is a permutation on the subscripts of s_i's and $(\theta^I(n-1), \ldots, \theta^I(1), \theta^I(0))$ is a permutation on the superscripts of d_0^i's.

In Fig. 2.5, we depict a 16×16 4-stage CP type MIN $\Gamma = \mathbf{P}^4\mathbf{E}^3\mathbf{P}^3 \cdots \mathbf{E}^0\mathbf{P}^0$, where $V^0 = (2,1,0,3)$, $V^1 = (2,3,0,1)$, $V^2 = (0,1,3,2)$, $V^3 = (2,0,3,1)$ and $V^4 = (1,3,0,2)$. The general form of transition sequences on Γ is as follows:

$$S = (s_3, s_2, s_1, s_0) \qquad D^0 = (s_2, s_1, s_0, s_3)$$

$$D^{00} = (s_2, s_1, s_0, d_2) \qquad D^1 = (s_1, s_2, d_2, s_0)$$

$$D^{11} = (s_1, s_2, d_2, d_3) \qquad D^2 = (d_3, d_2, s_1, s_2)$$

$$D^{22} = (d_3, d_2, s_1, d_0) \qquad D^3 = (d_2, d_0, d_3, s_1)$$

$$D^{33} = (d_2, d_0, d_3, d_1) \qquad D^4 = (d_3, d_2, d_1, d_0) = D\,.$$

We have $\mathbf{O}(s_3,s_2,s_1,s_0) = (s_1,s_2,s_0,s_3)$ and $(d_3,d_2,d_1,d_0) = \mathbf{I}(d^3,d^2,d^1,d^0) = (d^1,d^0,d^3,d^2)$, i.e., $V^O = (1,2,0,3)$ and $V^I = (1,0,3,2)$.

As we pointed out above, the combined effect of permuting bits of the operation sequence $\mathbf{P}^n \cdots \mathbf{P}^0$ on a CP$^{\min}$ type MIN Γ can be reflected by two permutation functions $\mathbf{O}$ and $\mathbf{I}$. The relationship between $\mathbf{O}$, $\mathbf{I}$, and the operation sequence $\mathbf{P}^n \cdots \mathbf{P}^0$ can be described by the following lemma.

LEMMA 1: On a CP$^{\min}$ type MIN Γ, the characteristic function $\mathbf{O}$ is uniquely determined by the sequence $\mathbf{P}^{n-1} \cdots \mathbf{P}^0$ and characteristic function $\mathbf{I}$ is uniquely determined by the sequence $\mathbf{P}^n \cdots \mathbf{P}^1$.

PROOF: Consider an arbitrary transition sequence of Γ. After the operation of $\mathbf{P}^i$, $i \in [0, n-1]$, bit $s_{\theta^O(i)}$ of input S is permuted to the position of LSB in D^i. Clearly, $\mathbf{P}^n$ has nothing to do with the function $\mathbf{O}$. Similarly, in $D^{(i)(i)}$, each $s_{\theta^O(i)}$ is replaced by bit d^i. Each d^i is then permuted by $\mathbf{P}^{i+1}$ and preserved in D^{i+1}. Thus, the final order of d^i's in D^n

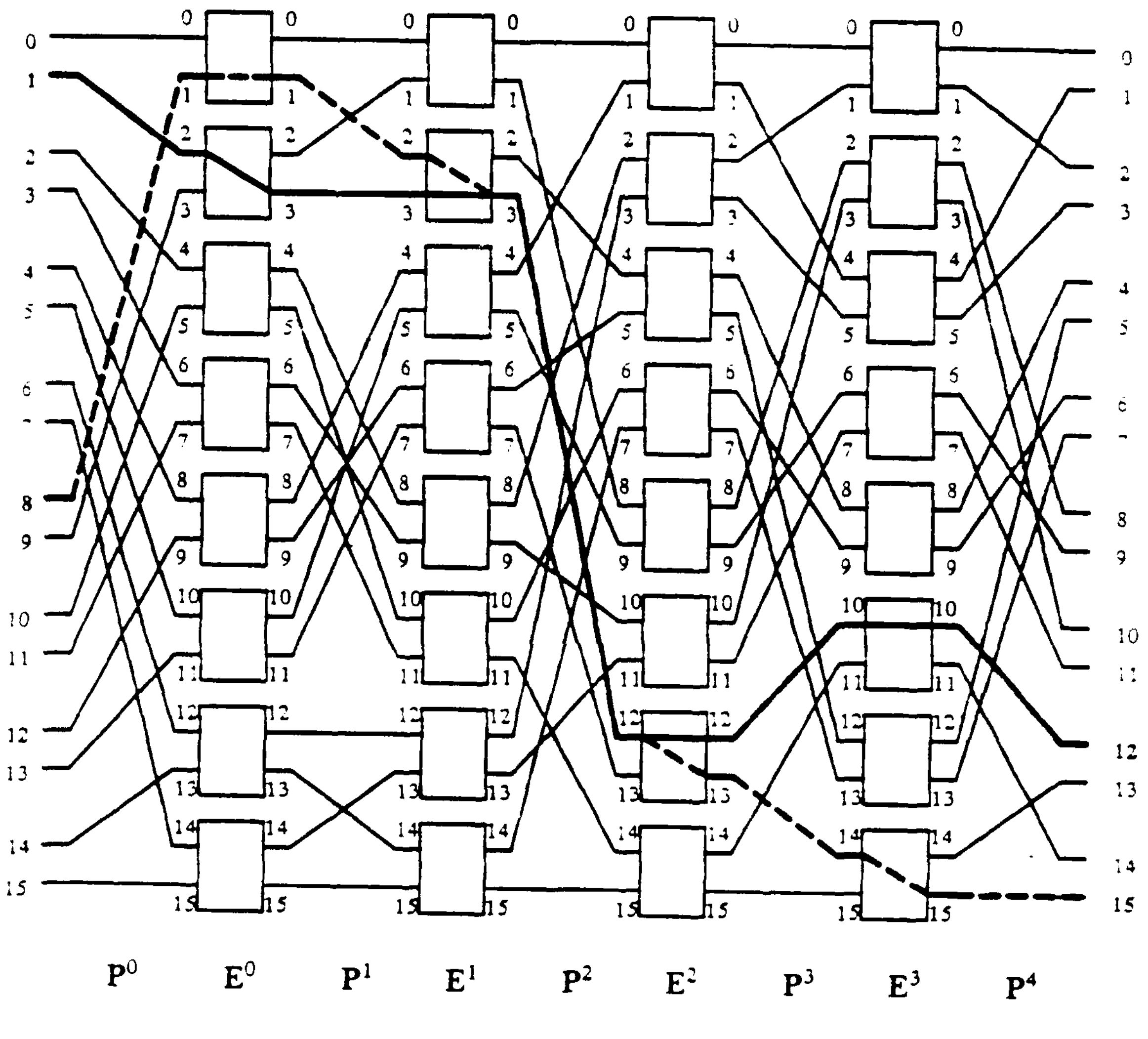


Fig. 2.5. A 16 × 16 4-stage CP^{min} type MIN are constructed with connection patterns specified by vectors V^0 = (2,1,0,3), V^1 = (2,3,0,1), V^2 = (0,1,3,2), V^3 = (2,0,3,1) and V^4 = (1,3,0,2). The conflicting path connections from S = (0,0,0,1) to D = (1,1,0,0) and A = (1,0,0,0) to B = (1,1,1,1) is also shown.

$= (d_0^{\theta(n-1)},\ldots,d_0^{\theta(1)},d_0^{\theta(0)})$, which can be specified by the function $\mathbf{I}$ has nothing to do with $\mathbf{P}^0$. $\square$

Now, we would like to investigate the topological characteristics of the reversal network of a CP$^{\min}$ type MIN. The reversal network Γ^R of a CP$^{\min}$ type MIN Γ is the network obtained from Γ by reversing the direction of each connection pattern, and replacing each input (output) by an output (input) without changing its label. Let $\mathbf{P}^{-(i)}$ denote the inverse function of a permutation function $\mathbf{P}^i \in \text{CP}(n)$. We have the following lemma on the topological structure of a reversal network.

LEMMA 2: The reversal network of a CP$^{\min}$ type MIN Γ defined as a permutation sequence $\mathbf{P}^n\mathbf{E}^{(n-1)}\mathbf{P}^{(n-1)} \cdots \mathbf{E}^0\mathbf{P}^0$ is a CP$^{\min}$ type MIN:

$$\Gamma^R = \mathbf{P}^{-(0)}\mathbf{E}^{(0)}\mathbf{P}^{-(1)} \cdots \mathbf{E}^{(n-1)}\mathbf{P}^{-(n-1)}.$$

Moreover, Γ^R has two characteristics functions $\mathbf{O}^R$ and $\mathbf{I}^R$ where $\mathbf{O}^R$ is uniquely determined by $\mathbf{P}^{-(1)}\mathbf{P}^{-(2)} \cdots \mathbf{P}^{-(n)}$ and $\mathbf{I}^R$ is uniquely determined by $\mathbf{P}^{-(0)}\mathbf{P}^{-(1)} \cdots \mathbf{P}^{-(n-1)}$.

PROOF. Since the reversal connection pattern of each connection pattern defined by $\mathbf{P}^i$ can be defined by the inverse function $\mathbf{P}^{-(i)}$ and the reversal switching stage of a switching stage is still the same switching stage, it is clear that the reversal network of Γ can be expressed as

$$\Gamma^R = \mathbf{P}^{-(0)}\mathbf{E}^{(0)}\mathbf{P}^{-(1)} \cdots \mathbf{E}^{(n-1)}\mathbf{P}^{-(n)}.$$

Besides, because of the isomorphism between graph models of Γ and Γ^R, Γ^R satisfies the Banyan property, i.e., Γ^R belongs to n-stage CP$^{\min}$. Thus, by a similar proof as that for Lemma 1, we can show that there exist two characteristic functions $\mathbf{O}^R$ and $\mathbf{I}^R$ such that $\mathbf{O}^R$ is uniquely determined by the sequence $\mathbf{P}^{-(1)}\mathbf{P}^{-(2)} \cdots \mathbf{P}^{-(n)}$ and $\mathbf{I}^R$ is uniquely determined by $\mathbf{P}^{-(0)}\mathbf{P}^{-(1)} \cdots \mathbf{P}^{-(n-1)}$.

□

In Table 2.1, the characteristic functions of several famous MINs and their reversal networks are summarized.

Three conclusions can be made for this section. First, it is very easy to observe that, except for $\mathbf{P}^0$ and $\mathbf{P}^n$, no other $\mathbf{P}^k$'s ($k \in [1, n\text{-}1]$) can be CP(n) permutation functions specified by $\theta(0) = 0$. This is because any $\mathbf{P}^k$ specified by $\theta^k(0) = 0$ in V^k will cause some bit s^i of input S to lose the chance to appear at the position of LSB in the transition sequence, and the n-stage MIN Γ will no longer preserve the Banyan property. This is a forbidden case described in Theorem 1. Second, as the equivalence problem is considered on the topologically equivalent class of CP$^{\text{min}}$, some subdivisions can be made. It is very easy to verify that there exists a one-to-many mapping between functions **O** or **I**, and MINs in the CP$^{\text{min}}$. The equivalent relationship between arbitrary MINs in the CP$^{\text{min}}$ can be classified as follows: catalogs:

The class of **O**-equivalent MINs: with the same function **O** but different function **I**.

The class of **I**-equivalent MINs: with the same function **I** but different function **O**.

The class of **O**/**I**-equivalent MINs: with the same function **I** and function **O**.

Therefore, the whole class of CP$^{\text{min}}$ can be partitioned into $(n\,!)^2$ **O**/**I**-equivalent classes. Each class with specified functions **O** and **I** has $[(n-1)!]^n$ different drawings. For example, it can be shown that a 4-stage CP$^{\text{min}}$ type MIN with connection patterns specifiea by $V^0 = (1,2,0,3)$, $V^1 = (2,3,0,1)$, $V^2 = (2,0,1,3)$, $V^3 = (1,0,2,3)$ and $V^4 = (1,3,0,2)$ is **O**/**I** equivalent to the MIN depicted in Fig. 2.5. Third, the equivalence relation in the CP$^{\text{min}}$ can easily be described by linear transformations between two arbitrary MINs. Assume that two CP$^{\text{min}}$ type MIN, Γ_1

Table 2.1. The characteristic functions of several famous MINs and their reversal networks.

MIN's	V^O	V^I
Delta network	$(1,2,\ldots,n-1,0)$	$(0,\ldots,n-2,n-1)$
reversal Delta network	$(n-1,\ldots,1,0)$	$(0,n-1,\ldots,2,1)$
Omega network	$(0,\ldots,n-2,n-1)$	$(0,\ldots,n-2,n-1)$
reversal Omega network	$(n-1,\ldots,1,0)$	$(n-1,\ldots,1,0)$
Baseline network	$(n-1,\ldots,1,0)$	$(0,\ldots,n-2,n-1)$
reversal Baseline network	$(n-1,\ldots,1,0)$	$(0,\ldots,n-2,n-1)$
Indirect Binary Cube network	$(n-1,\ldots,1,0)$	$(n-1,\ldots,1,0)$
reversal Indirect Binary Cube network	$(0,\ldots,n-2,n-1)$	$(0,\ldots,n-2,n-1)$

and Γ_1, are specified by functions $\mathbf{O}_1$, $\mathbf{I}_1$ and $\mathbf{O}_2$, $\mathbf{I}_2$, respectively. If we transform Γ_1 to Γ_2, we can always have the followi:g relation: there exist two functions $\mathbf{F}$, $\mathbf{R} \in \mathrm{CP}(n)$ such that

$$\mathbf{O}_2\mathbf{F} = \mathbf{O}_1 \quad \text{and} \quad \mathbf{R}\mathbf{I}_2 = \mathbf{I}_1,$$

or

$$\mathbf{F} = \mathbf{O}_2^{-1}\mathbf{O}_1 \quad \text{and} \quad \mathbf{R} = \mathbf{I}_1\mathbf{I}_2^{-1}.$$

Functions $\mathbf{F}$ and $\mathbf{R}$ represent two linear transformations or two fixed connection patterns added before the first and after the last connection pattern of Γ_2. They can also be interpreted as the renaming scheme on the inputs and outputs of Γ_2, i.e., the inputs are renamed according to function $\mathbf{F}^{-1}$ and the outputs, function $\mathbf{R}^{-1}$. The renaming scheme can easily transform one network to another network without any hardware cost. It has a significant impact on designing reconfigurable systems.

2.4. ROUTING ALGORITHM

In a general MIN, routing is established by attaching to each input a path control sequence or a *path descriptor* [KrSn86] to lead it to a desired output. Generally speaking, paths from different inputs to the same output may have different control sequences. Thus, a routing table, containing a path control sequence for each output, is needed at each input. From the viewpoint of simple routing, it is convenient to have all these tables identical.

As discussed in previous sections, any path in a $\mathrm{CP}^{\mathrm{min}}$ type MIN leading from an input to an output can be represented by a transition sequence. Note that each D^i and $D^{(i)(i)}$ in a transition sequence represent the address of input and output links through which a path traverses stage i. Moreover, the ordered set of all the LSB's in D^i's $(s_0^{n-1}, \cdots, s_0^1, s_0^0)$

which equals $(s_{\theta^o(n-1)},...,s_{\theta^o(1)},s_{\theta^o(0)})$ is a permutation on the bits of the address of an input. Similarly, the binary representation of an output $(d_{n-1},...,d_1,d_0)$ which equals $(d^{\theta^I(n-1)},...,d^{\theta^I(1)},d^{\theta^I(0)})$ is a mapping of a permutation on the ordered set of all the LSB's in $D^{(i)(i)}$'s $(d^{n-1}, . . . , d^1,d^0)$. Therefore, three conclusions can be made on a CP^{min} type MIN. First, any path connecting an input and an output preserves the information of input and output addresses, and indicates the topological structure of this MIN reflected by functions **O** and **I**. Second, the LSB's in D^i's and $D^{(i)(i)}$'s represent labels of input and output links in a switching element (i.e., 0-input (0-output) or 1-input (1-output)) through which a path traverses stage i. Hence, at each stage, regardless of the input link through which a path traverses, this path can always be routed to the desired output link (i.e., $s_{\theta^o(i)}$ is replaced by d^i which should be a binary bit of D). It is natural that each path control sequence can be constructed by using only the address of a destination output. In this chapter, this is referred to as destination-oriented. Third, since the content of a path control sequence is only related to the destination output to which a source input desires to route and the function **I**, all the routing tables are identical. This is required by a simple routing scheme.

Thus, briefly speaking, the distributed routing on an n-stage CP^{min} type MIN Γ which is characterized by **O** and **I** is accomplished by the source input attached with the path control sequence $T = (t_{n-1}, . . . , t_1,t_0)$ as routing tags along with a request for connection. As the request progresses through the stages of Γ, the switching element at stage i uses the tag t_i from the path control sequence T to route the incoming request via the particular output link determined by t_i, i.e., via 0-output link if $t_i = 0$ and 1-output link if $t_i = 1$. Eventually, the request reaches the correct destination. In other words, the distributed routing can be accomplished under local control at each switching element. Any switching element is said to be in the 1-state if a crossing connection (from 0-input to 1-output or from 1-input to 0-output) has been established and in the 0-state if a straight connection (from 0-input to 0-output or from 1-input to 1-output) has been established. Let l_i be the label of the input from which an incoming request comes and l_o be the label of the output link to which the routing tag determines to route. Obviously, the state of a switching element can be obtained by performing an Exclusive-OR on l_i and l_o. Next, we study the general form of the routing algorithm for the $\log_2 N$ $\mathrm{CP}^{\mathrm{min}}$. Let **XOR** be the modulo 2 addition and $S \rightarrow D$ denote the path connection from input $S = (s_{n-1}, \ldots, s_1, s_0)$ to output $D = (d_{n-1}, \ldots, d_1, d_0)$.

THEOREM 2: Let $T = (t_{n-1}, \ldots, t_1, t_0)$ be the path control sequence of $S \rightarrow D$ on an n-stage $\mathrm{CP}^{\mathrm{min}}$ type MIN $\mathbf{\Gamma}$. Then t_i has the following form:

$$t_i = \mathbf{\Gamma}^{-1}(d_i).$$

Moreover, $g_i = \mathrm{O}(s_i)$ **XOR** $\mathbf{\Gamma}^{-1}(d_i)$ can determine the state of the switching element through which $S \rightarrow D$ traverses at stage i. $G = (g_{n-1}, \ldots, g_1, g_0)$ is called the *path state sequence*.

PROOF: As we remarked above that the path $S \rightarrow D$ can always be routed via $\mathrm{O}(s_i)$ or the $s_{\theta^o(i)}$-input link of some switching element at stage i, the selection of a correct output link at stage i (i.e., the replacement operation performed on to $\mathrm{O}(s_i)$) is clearly irrelevant to the incoming $\mathrm{O}(s_i)$-input link. Therefore, as long as T is used as the path control sequence, it is equivalent to saying that $S \rightarrow D$ traverses some switching element at stage i from an $\mathrm{O}(s_i)$-input link to an output link whose label is $\mathbf{\Gamma}^{-1}(d_i)$, i.e., $d^i = d_{\theta^{j-1}(i)}$. Thus, we have the following transition sequence:

$$S = (s_{n-1}, \ldots, s_1, s_0)$$

$$D^0 = (s_{n-1}^0, \ldots, s_1^0, s_{\theta^o(0)})$$

$$D^{(0)(0)} = (s_{n-1}^0, \ldots, s_1^0, d^0)$$

......

$$D^n = (d^{\theta'(n-1)}, \ldots, d^{\theta'(1)}, d^{\theta'(0)})$$

$$= (d_{\theta'(\theta'^{-1}(n-1))}, \ldots, d_{\theta'(\theta'^{-1}(1))}, d_{\theta'(\theta'^{-1}(0))})$$

$$= (d_{n-1}, \ldots, d_1, d_0).$$

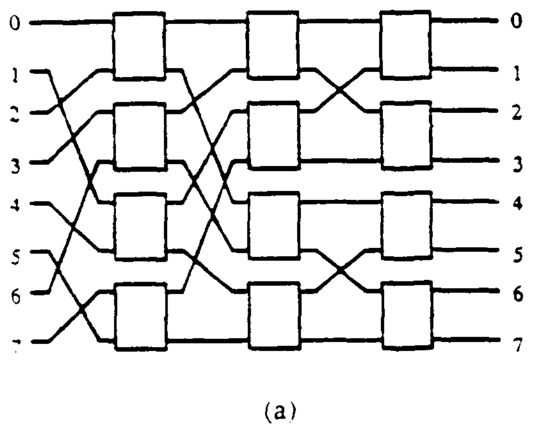
Obviously, it is valid and $\mathrm{I}^{-1}(d_i)$ is the only possible routing tag which can be used at stage i. However, the state of the switching element at stage i is determined by $\mathrm{O}(s_i)$ **XOR** $\mathrm{I}^{-1}(d_i)$. □

For example, the path control sequences and path state sequences of the famous MINs in Table 2.1 are summarized in Table 2.2. For the MIN in Fig. 2.5, we have $\mathrm{I}^{-1}(D) = (d_1, d_0, d_3, d_2)$. The path connection from $S = (0,0,0,1)$ to $D = (1,1,0,0)$ is shown by a **bold** line, where from Theorem 2 we can get $T = (0,0,1,1)$ and $G = (0,0,0,1)$.

Theorem 2 provides an efficient routing scheme on a packet-switching $\mathrm{CP}^{\mathrm{min}}$ type MIN. In particular, in packet-switching networks, when messages are sent from inputs to outputs, replies are returned to the sender. Thus, the address of the sender is needed. Instead of attaching the sender address to a message, the address can be created while passing the MIN: whenever bit $\mathrm{I}^{-1}(d_i)$ in the path control sequence is discarded at stage i, it is replaced by bit $\mathrm{O}(s_i)$ that identifies the input link from which the message came from. Eventually, this message preserves the address of the sender as it arrives at the receiver. We should note that not every Banyan type MIN has this natural property. For example, in Fig. 2.6, two 3-stage non-$\mathrm{CP}^{\mathrm{min}}$ Banyan type MINs are shown. The MIN in Fig. 2.6(a) is topologically equivalent to $\mathrm{CP}^{\mathrm{min}}$, but the other one in Fig. 2.6(b) is not. Even if their routing schemes are also

Table 2.2. The path control sequences and path state sequences of those MINs in Table 2.1.

MIN's	$T = (t_{n-1}, \ldots, t_1, t_0)$	$G = (g_{n-1}, \ldots, g_1, g_0)$
Delta network	$(d_0, \ldots, d_{n-2}, d_{n-1})$	$(s_1 \oplus d_0, s_2 \oplus d_1, \ldots, s_{n-1} \oplus d_{n-2}, s_0 \oplus d_{n-1})$
reversal Delta network	$(d_0, d_{n-1}, \ldots, d_2, d_1)$	$(s_{n-1} \oplus d_0, s_{n-2} \oplus d_{n-1}, \ldots, s_1 \oplus d_2, s_0 \oplus d_1)$
Omega network	$(d_0, \ldots, d_{n-2}, d_{n-1})$	$(s_0 \oplus d_0, \ldots, s_{n-2} \oplus d_{n-2}, s_{n-1} \oplus d_{n-1})$
reversal Omega network	$(d_{n-1}, \ldots, d_1, d_0)$	$(s_{n-1} \oplus d_{n-1}, \ldots, s_1 \oplus d_1, s_0 \oplus d_0)$
Baseline network	$(d_0, \ldots, d_{n-2}, d_{n-1})$	$(s_{n-1} \oplus d_0, \ldots, s_1 \oplus d_{n-2}, s_0 \oplus d_{n-1})$
reversal Baseline network	$(d_0, \ldots, d_{n-2}, d_{n-1})$	$(s_{n-1} \oplus d_0, \ldots, s_1 \oplus d_{n-2}, s_0 \oplus d_{n-1})$
Indirect Binary Cube network	$(d_{n-1}, \ldots, d_1, d_0)$	$(s_{n-1} \oplus d_{n-1}, \ldots, s_1 \oplus d_1, s_0 \oplus d_0)$
reversal Indirect Binary Cube network	$(d_0, \ldots, d_{n-2}, d_{n-1})$	$(s_0 \oplus d_0, \ldots, s_{n-2} \oplus d_{n-2}, s_{n-1} \oplus d_{n-1})$



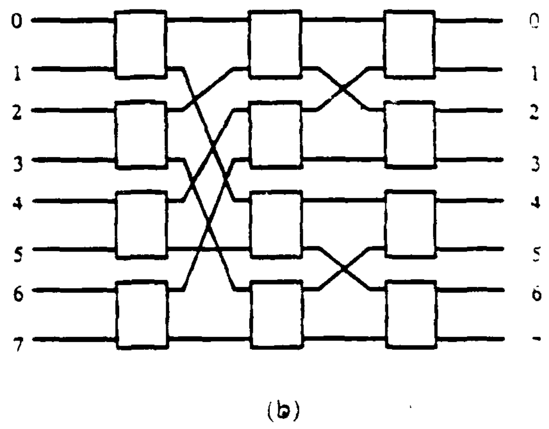


Fig. 2.6. Two 3-stage *non* - CP^{min} Banyan type MINs are shown.

destination-oriented, there is no simple rule to preserve the sender address on them.

In a CP^{min} type MIN, two path connections which do not result in connection conflicts, (i.e., they do not use the same output link at some switching element) are said to be *conflict-free*. The next theorem is a necessary and sufficient condition for two conflict-free path connections on the class of $\log_2 N$ CP^{min}. It is the extension of the theorem in [WuFe81] which only deals with the Baseline network.

DEFINITION 4: Let S and M be two n-bit numbers. $\phi(S,A)$ yields the maximum number of consecutively identical low-order bits of S and A. $\psi(S,A)$ yields the maximum number of consecutively identical high-order bits of S and A. □

For example, if $S = (0,1,1,0,1,0)$ and $A = (0,1,0,0,1,0)$, we have $\phi(S,A) = 3$ and $\psi(S,A) = 2$.

THEOREM 3: In an n-stage CP^{min} type MIN characterized by **O** and **I**, two path connections $S \rightarrow D$ and $A \rightarrow B$, where $S \neq A$ and $D \neq B$, are conflict-free if and only if

$$\psi(\, \mathbf{O}(S),\mathbf{O}(A)\,) + \phi(\, \mathbf{I}^{-1}(D),\mathbf{I}^{-1}(B)\,) < n .$$

PROOF: (only if) Suppose $\psi(\, \mathbf{O}(S),\mathbf{O}(A)\,) = k$ and $\phi(\, \mathbf{I}^{-1}(D),\mathbf{I}^{-1}(B)\,) = k'$. According to Theorem 4, bit $\mathbf{O}(s_i)$ on the transition sequence is replaced by bit $\mathbf{I}^{-1}(d_i)$ after $S \rightarrow D$ traversing stage i. Thus, generally speaking, at stage i, $0 \leq i \leq n-1$, the output links traversed by $S \rightarrow D$ and $A \rightarrow B$ are $\Pi(\mathbf{O}(s_{n-1}), \ldots, \mathbf{O}(s_{i+1}), \mathbf{I}^{-1}(d_i), \ldots, \mathbf{I}^{-1}(d_0))$ and $\Pi(\mathbf{O}(a_{n-1}), \ldots, \mathbf{O}(a_{i+1}), \mathbf{I}^{-1}(b_i), \ldots, \mathbf{I}^{-1}(b_0))$ where $\Pi \in CP(n)$. $\Pi(\mathbf{O}(s_{n-1}), \ldots, \mathbf{O}(s_{i+1}), \mathbf{I}^{-1}(a_i), \ldots, \mathbf{I}^{-1}(d_0))$ and $\Pi(\mathbf{O}(a_{n-1}), \ldots, \mathbf{O}(a_{i+1}), \mathbf{I}^{-1}(b_i), \ldots, \mathbf{I}^{-1}(b_0))$ are those $D^{(i)(i)}$'s on the transition sequences of $S \rightarrow D$ and $A \rightarrow B$, respectively. By the definition of ϕ, whenever $1 \leq i \leq k'$, we have $(\mathbf{I}^{-1}(d_i), \ldots, \mathbf{I}^{-1}(d_0)) = (\mathbf{I}^{-1}(b_i), \ldots, \mathbf{I}^{-1}(b_0))$. Since

$\psi(O(S), O(A)) = k$ and $k + k' < n$, we must have $k < n-k' \le n-i-1$ and $(O(s_{n-1}), \ldots, O(s_{i+1})) \neq (O(a_{n-1}), \ldots, O(a_{i+1}))$. Similarly, if $k' < i < n$, then $(I^{-1}(d_i), \ldots, I^{-1}(d_0)) \neq (I^{-1}(b_i), \ldots, I^{-1}(b_0))$. Consequently,

$$\Pi(O(s_{n-1}), \ldots, O(s_{i+1}), I^{-1}(d_i), \ldots, I^{-1}(d_0)) \neq$$

$$\Pi(O(a_{n-1}), \ldots, O(a_{i+1}), I^{-1}(b_i), \ldots, I^{-1}(b_0))$$

at each stage i, $0 \le i \le n-1$. Therefore, $S \to D$ and $A \to B$ are conflict-free.

(if) Since $S \to D$ and $A \to B$ are conflict-free, we have

$$\Pi(O(s_{n-1}), \ldots, O(s_{i+1}), I^{-1}(d_i), \ldots, I^{-1}(d_0)) \neq$$

$$\Pi(O(a_{n-1}), \ldots, O(a_{i+1}), I^{-1}(b_i), \ldots, I^{-1}(b_0))$$

at each stage i, $0 \le i \le n-1$. This implies that $(O(s_{n-1}), \ldots, O(s_{i+1})) \neq (O(a_{n-1}), \ldots, O(a_{i+1}))$ or $(I^{-1}(d_i), \ldots, I^{-1}(d_0)) \neq (I^{-1}(b_i), \ldots, I^{-1}(b_0))$, at each stage i. As a result, $\psi(\ O(S), O(A)\ ) + \phi(\ I^{-1}(D), I^{-1}(B)\ ) < n$. $\square$

For example, in Fig. 5, if $S = (0,0,0,1)$, $D = (1,1,0,0)$, $A = (1,0,0,0)$ and $B = (1,1,1,1)$, we have $O(S) = (0,0,1,0)$, $I^{-1}(D) = (0,0,1,1)$, $O(A) = (0,0,0,1)$, and $I^{-1}(B) = (1,1,1,1)$. Thus, $\psi(\ O(S), O(A)\ ) = 2$ and $\phi(\ I^{-1}(D), I^{-1}(B)\ ) = 2$. As a result of Theorem 4, $S \to D$ and $A \to B$ are two conflicting path connections.

2.5. DECOMPOSITION AND PARTITIONING

In this section, compared to the distributed routing property, we study two global functional properties on the CP^{min}, i.e., its *decomposition* and *partitioning*.

The *partitionability* of a network is the ability to decompose the network into independent subnetworks of different sizes [Sie79]. It is desirable that each subnetwork with smaller size can have all the functional properties of the original network. A partitionable network allows a system to be dynamically reconfigured into independent subsystems. It has a significant impact on the application of parallel processing.

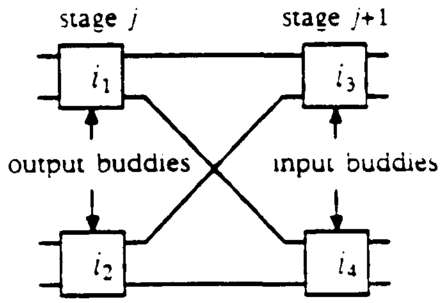
In the next theorem, we exploit the *strict buddy property* [Agr83] of the class of $\log_2 N$ CP^{min}. It is an important auxiliary for the discussion on decomposition and partitioning.

Let Y_1 and Y_2 be two switching elements at a switching stage as discussed in Section II. Y_1 and Y_2 are said to be *output buddies* if $succ^0(Y_1) = succ^0(Y_2)$ and $succ^1(Y_1) = succ^1(Y_2)$; *input buddies* if $prec^0(Y_1) = prec^0(Y_2)$ and $prec^1(Y_1) = prec^1(Y_2)$. A MIN has the strict buddy property if and only if at each stage for each pair of input buddies there exists another pair of input buddies such that they constitute two pairs of output buddies. Fig. 2.7 illustrates the buddy property.

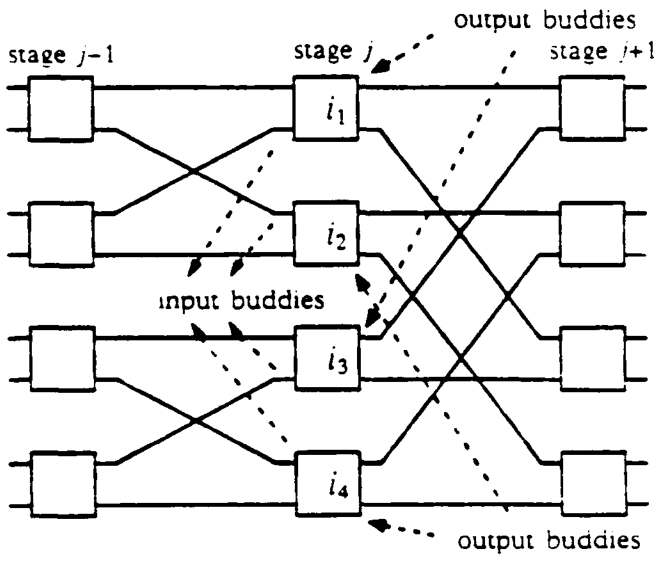
THEOREM 4: The class of n-stage CP^{min} satisfies the strict buddy property.

PROOF: Let Γ be an n-stage CP^{min} type MIN with characteristic functions **O** and **I** as we have defined. Even if the function **O** cannot uniquely determine the topological structure of Γ, without loss of generality, we can construct Γ by using only, say, the k-subshuffle σ_k. If other CP(n) type connection patterns are used, the proof is similar to the following.

Let $SW_{x,j}$ be the xth switching element at stage j of Γ, where $x \in [0, \frac{N}{2}-1]$, $j \in [0, n-1]$, and its binary representation be $(w_{n-1}, \ldots, w_1)$. Assume stage j is connected to stage $j+1$ by a σ_k connection pattern. The labels of the successors at stage $j+1$ of $SW_{x,j}$ are of the form $(w_{n-1}, \ldots, w_{k+1}, w_{k-1}, \ldots, w_1, d)$, where $d = 0$ or 1. It is obvious that we can always find a unique switching element $SW_{y,j}$ which has the same successors as $SW_{x,j}$ if and only if its label is $(w_{n-1}, \ldots, w_{k+1}, \bar{w}_k, w_{k-1}, \ldots, w_1)$, where $\bar{w}_k$ represents the binary complement



(a)



(b)

Fig. 2.7 (a) The buddy property . (b) The Interstage buddy property in an MIN.

of w_k. Therefore, $SW_{x,j}$ and $SW_{y,j}$ are output buddies at stage j and their common successors are input buddies at stage $j+1$. Thus, each input buddies and output buddies can easily be identified at each stage of Γ.

Now, to show the strict buddy property of Γ, we must verify that for each pair of input buddies at stage $j+1$ there exists another pair of input buddies such that they constitute two pairs of output buddies.

Let $SW_{x,j+1}$, $SW_{y,j+1}$ and $SW_{u,j+1}$, $SW_{v,j+1}$ be two pairs of input buddies and let $(a_{n-1}, \ldots, a_{k+1}, a_{k-1}, \ldots, a_{l+1}, a_l, a_{l-1}, \ldots, a_1, d)$, and $(b_{n-1}, \ldots, b_{k+1}, b_{k-1}, \ldots, b_{l+1}, b_l, b_{l-1}, \ldots, b_1, d)$ be their labels, where $d = 0$ corresponds to $SW_{x,j+1}$, $SW_{u,j+1}$; $d = 1$ corresponds to $SW_{y,j+1}$, $SW_{v,j+1}$. Assume stage $j+1$ is connected to stage $j+2$ through a σ_{l+1} connection pattern. If we let $a_i = b_i = c_i$, $i \in [0, l\text{-}1] \bigcup [l+1, k\text{-}1] \bigcup [k+1, n\text{-}1]$ and $a_l = \bar{b}_l$, $SW_{x,j+1}$ and $SW_{u,j+1}$ will form output buddies with common successors $(c_{n-1}, \ldots, c_{k+1}, c_{k-1}, \ldots, c_{l+1}, c_{l-1}, \ldots, c_1, 0, d)$ at stage $j+2$. Similarly, $SW_{y,j+1}$ and $SW_{v,j+1}$ will form output buddies with common successors $(c_{n-1}, \ldots, c_{k+1}, c_{k-1}, \ldots, c_{l+1}, c_{l-1}, \ldots, c_1, 1, d)$ at stage $j+2$. Also it is clearly that this is the only possible way to specify two pairs of input buddies which also constitute two pairs of output buddies. Hence, at any switching stage, for each input buddies the uniqueness of selecting another input buddies to constitute two pairs of output buddies is verified. The argument is also applicable to the boundary switching stages, stage 0 and stage n-1, by imagining inputs (outputs) of Γ to be output (input) links of a pseudo switching stage. Therefore, Γ satisfies the strict buddy property. $\square$

Now, we discussion the decomposition property on the class of n-stage CP^{min}.

THEOREM 5: An n-stage CP^{min} type MIN Γ can be decomposed from the view of each stage j, $j \in [0, n-1]$, as follows (see Fig. 2.8):

(*Forward*) Stage j is followed by two disjoint $(n-j-1)$-stage subnetworks, $\mathbf{N}_0$ and $\mathbf{N}_1$, such that each switching element on stage j is connected to an input of $\mathbf{N}_0$ ($\mathbf{N}_1$) via its 0 (1)-output link.

(*Backward*) Stage j is leaded by two disjoint j-stage subnetworks, $\mathbf{M}_0$ and $\mathbf{M}_1$, such that each switching element on stage j is connected to an output of $\mathbf{M}_0$ ($\mathbf{M}_1$) via its 0 (1)-input link.

PROOF: As in Theorem 6, without loss of generality, we assume only subshuffle connection patterns are used on Γ. Also we assume stage j is connected to stage $j+1$ by σ_k connection pattern and stage $j+1$ is connected to stage $j+2$ by σ_{l+1} connection pattern. Consider the forward case first. It is clear that, on stage j and stage $j+1$, each 0-output link at stage j with label $(w_{n-1},...,w_1,0)$ is connected to a switching element at stage $j+1$ with label $(w_{n-1},...,w_{k+1},w_{k-1},...,w_1,0)$; and each 1-output link at stage j with label $(w_{n-1},...,w_1,1)$ is connected to a switching element at stage $j+1$ with label $(w_{n-1},...,w_{k+1},w_{k-1},...,w_1,1)$. Thus, each switching element at stage j is connected to a set of 2^{n-1} switching elements, S_0, with general label form $(d, \ldots, d, 0)$ at stage $j+1$ via its 0-output link. In a similar way, each switching element at stage j is connected to a set of 2^{n-1} switching elements, S_1, with general label form $(d, \ldots, d, 1)$ at stage $j+1$ via its 1-output link.

Now, we want to show that each one of S_0 and S_1 can span an $(n-j-1)$-stage subnetwork and, particularly, these two spanned subnetworks are disjoint. For any switching element $(x_{n-1},...,x_{k+1},x_{k-1},...,x_{l+1},x_l,x_{l-1},...,x_1,0) \in S_0$, its corresponding switching element

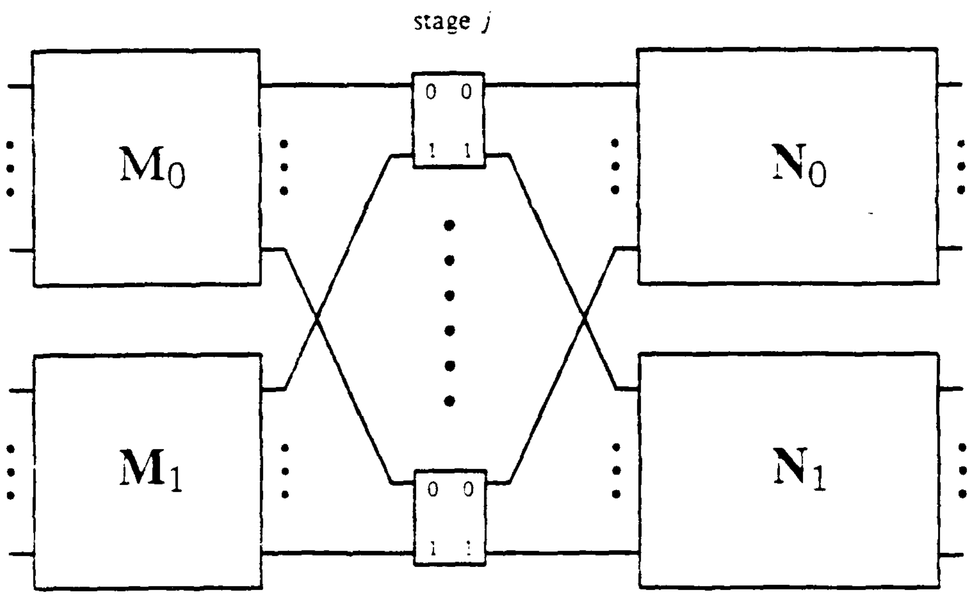


Fig. 2.8. The decomposition property of a CP^{min} type MIN.

which has the label $(x_{n-1},...,x_{k+1},x_{k-1},...,x_{l+1},\bar{x}_l,x_{l-1},...,x_1,0)$ to form output buddies is still an element in S_0. There are a total of 2^{n-2} pairs of output buddies in S_0 which can span a set of 2^{n-1} switching elements consisting of 2^{n-2} pairs of input buddies at stage $j+2$. Each output buddies in S_0 with label form $(x_{n-1}, \ldots, x_{k+1}, x_{k-1}, \ldots, x_{l+1}, d, x_{l-1}, \ldots, x_1, 0)$ spans input buddies with label form $(x_{n-1}, \ldots, x_{k+1}, x_{k-1}, \ldots, x_{l+1}, x_{l-1}, \ldots, x_1, 0, d)$ at stage $j+2$. According to Theorem 3 and the definition of CP(n) type connection patterns, some bit x_i in label $(x_{n-1}, \ldots, x_{k+1}, x_{k-1}, \ldots, x_{l+1}, x_{l-1}, \ldots, x_1, 0, d, d)$ is permuted to the position of LSB. $(x_{n-1}, \ldots, x_{k+1}, x_{k-1}, \ldots, x_{l+1}, x_{l-1}, \ldots, x_1, 0, d, d)$ is the label form of output links of the set of switching elements at stage $j+2$ composed of input buddies $(x_{n-1}, \ldots, x_{k+1}, x_{k-1}, \ldots, x_{l+1}, x_{l-1}, \ldots, x_1, 0, d)$. Thus, we can have another set of 2^{n-1} switching elements spanned by S_0 at stage $j+3$ with the label form $(...,0,d,d)$. As we proved in Theorem 6, this is essentiality of the strict buddy property, because the corresponding input buddies of each input buddies at stage $j+2$ as required to satisfy strict buddy property has the same label form. In general, at each stage i, $i \in [j+2, n-1]$, S_0 spans a set of 2^{n-1} switching elements with general label form $(...,0,d,d,...,d)$ where the number of d's is equal to i. This is based on the fact of the strict buddy property at each stage of Γ. Hence, S_0 spans an $(n-j-1)$-stage subnetwork with 2^{n-1} inputs and outputs. We denote it by $\mathbf{N}_0$.

Similarly, S_1 spans a set of 2^{n-1} switching elements at each stage i, $i \in [j+2, n-1]$, with general label form $(...,1,d,d,...,d)$ where the number of d is equal to i. The $(n-j-1)$-stage subnetwork spanned by S_1 is denoted by $\mathbf{N}_1$. Obviously, $\mathbf{N}_0$ and $\mathbf{N}_1$ are disjoint, since their switching elements at each stage belong to two different sets.

For the backward case, the proof is similar. Again, based on the strict buddy property of Γ, two disjoint j-stage subnetworks, $\mathbf{M}_0$ and $\mathbf{M}_1$, can be found. Each switching element on stage j is connected to an output of $\mathbf{M}_0(\mathbf{M}_1)$ via its 0 (1)-input link. $\square$

The last theorem implies an alternative, recursive definition of CP^{min}. If Γ is decomposed as viewed from stage 0 (stage n-1), $\mathbf{N}_0$ and $\mathbf{N}_1$ ($\mathbf{M}_0$ and $\mathbf{M}_1$) will be two (n-1)-stage subnetworks. We can prove that they belong to the (n-1)-stage CP^{min} and, thus, can be decomposed in a recursive way.

LEMMA 3: An n-stage CP^{min} type MIN Γ can be decomposed as viewed from stage 0 (stage n-1) such that stage 0 (stage n-1) is followed by two disjoint (n-1)-stage CP^{min} type MIN's, $\mathbf{N}_0$ and $\mathbf{N}_1$ ($\mathbf{M}_0$ and $\mathbf{M}_1$).

PROOF: Without loss of generality, we assume only subshuffle connection patterns are used on Γ and a k-subshuffle σ_k is used at stage 0. According to Theorem 7, we have $\mathbf{N}_0$ and $\mathbf{N}_1$, two disjoint (n-1)-stage MIN's connected to those 0-output links and 1-output links at stage 0. We want to prove that they are two CP^{min} type MIN's.

As per the proof in Theorem 7, those input addresses associate with $\mathbf{N}_0$ have the form

$$S^{N_0} = D^{(0)(0)} = (s_{n-1}, \ldots, s_{k+1}, s_{k-1}, \ldots, s_0, 0)$$

$$= (s_{n-2}^{N_0}, \ldots, s_1^{N_0}, s_0^{N_0}, 0)$$

which is the label form of 0-output links at stage 0. Similarly, those input addresses associate with $\mathbf{N}_1$ have the form

$$S^{N_1} = D^{(0)(0)} = (s_{n-1}, \ldots, s_{k+1}, s_{k-1}, \ldots, s_0, 1)$$

$$= (s_{n-2}^{N_1}, \ldots, s_1^{N_1}, s_0^{N_1}, 1)$$

which is the label form of 1-output links at stage 0. We know that Γ is characterized by function $\mathbf{O}$ such that

$$\mathbf{O}(s_{n-1}, \ldots, s_{k+1}, s_k, s_{k-1}, \ldots, s_0) = (s_{\theta^o(n-1)}, \ldots, s_{\theta^o(1)}, s_{\theta^o(0)})$$

$$= (s_{\theta^o(n-1)}, \ldots, s_{\theta^o(1)}, s_k).$$

Thus, the change of s_k is irrelevant to any partial transition sequence starting from S^{N_0} (S^{N_1}) on $\mathbf{N}_0$ ($\mathbf{N}_1$). Consequently, there exists a restricted function $\mathbf{O}^N$ such that

$$\mathbf{O}^N(s_{n-1}, \ldots, s_{k+1}, s_{k-1}, \ldots, s_0)$$

$$= \mathbf{O}^N(s_{n-2}^{N_0}, \ldots, s_1^{N_0}, s_0^{N_0})$$

$$= \mathbf{O}^N(s_{n-2}^{N_1}, \ldots, s_1^{N_1}, s_0^{N_1})$$

$$= (s_{\theta^o(n-1)}, \ldots, s_{\theta^o(1)}).$$

That is to say, according to Theorem 3, $\mathbf{N}_0$ and $\mathbf{N}_1$ belong to the (n-1)-stage CP$^{\min}$.

Similarly, we can show that $\mathbf{M}_0$ and $\mathbf{M}_1$ are two disjoint (n-1)-stage CP$^{\min}$ type MIN's.□

In general, by properly linking decomposed subnetworks, each MIN in the CP$^{\min}$ can be decomposed recursively from the view of each stage. We have given the special cases at the first stage and the last stage in the above lemma. An efficient partition scheme can be achieved on the CP$^{\min}$ based on its topological decomposition property.

THEOREM 6: An n-stage CP$^{\min}$ type MIN Γ is decomposed to $\mathbf{N}_0$, $\mathbf{N}_1$, $\mathbf{M}_0$ and $\mathbf{M}_1$ four subnetworks from the view of stage k according to Theorem 7. If we force all switching elements at stage k to 0-state or 1-state, then $\mathbf{M}_i$ and $\mathbf{N}_j$, $i,j \in [0, 1]$, can be linked into an

$(n\text{-}1)$-stage CP$^{\text{min}}$ type MIN, denoted by $\mathbf{J}_{ij}$.

PROOF: This proof is similar to the last one. By forcing all the switching elements at stage k to 0-state (i.e. $s_{\theta^o(k)}$ is replaced by $s_{\theta^o(k)}$ in $D^{(k)(k)}$ on any transition sequence), we can link $\mathbf{M}_0$ and $\mathbf{N}_0$ ($\mathbf{M}_1$ and $\mathbf{N}_1$) to form an $(n\text{-}1)$-stage MIN. They are denoted as $\mathbf{J}_{00}$ and $\mathbf{J}_{11}$, respectively. Similarly, two $(n\text{-}1)$-stage MIN's, $\mathbf{J}_{01}$ and $\mathbf{J}_{10}$, can be formed by forcing all the switching elements at stage k to 1-state (i.e. $s_{\theta^o(k)}$ is replaced by $\bar{s}_{\theta^o(k)}$ in $D^{(k)(k)}$ on any transition sequence). The change of $s_{\theta^o(k)}$ is irrelevant to any joined partial transition sequence on $\mathbf{J}_{00}$, $\mathbf{J}_{11}$, $\mathbf{J}_{01}$ and $\mathbf{J}_{10}$. Therefore, all the $\mathbf{J}_{ij}$'s are $(n\text{-}1)$-stage CP$^{\text{min}}$ type MIN. □

LEMMA 4: An n-stage CP$^{\text{min}}$ type MIN Γ can be partitioned into two $(n-1)$-stage CP$^{\text{min}}$ type MIN's by forcing all the switching elements at stage i to 0-state or 1-stage such that, in each $(n-1)$-stage MIN, all the input addresses agree in bit $\mathrm{O}(s_i)$ and all the output addresses agree in bit $\mathrm{I}^{-1}(d_i)$.

PROOF: It is very easy to give the proof on a transition sequence using our path-descriptive methodology. To force all switching elements at stage i to 0-state (1-state) is equivalent to forcing the LSB $\mathrm{O}(s_i)$ or $s_{\theta^o(i)}$ of $D_{(i)}$ to be replaced by $s_{\theta^o(i)}$ ($\bar{s}_{\theta^o(i)}$) as the result in $D^{(i)(i)}$ on each transition sequence on Γ. First, we consider the 0-stage case. As per the nature of the routing scheme on Γ, at stage i, any message is routed from $\mathrm{O}(s_i)$-input link to $\mathrm{I}^{-1}(d_i)$-output link. That is to say, any input S with bit $\mathrm{O}(s_i)$ can communicate with any output D with bit $\mathrm{I}^{-1}(d_i) = \mathrm{O}(s_i)$. Obviously, we have divided Γ to two subnetworks. According to Theorem 7 and Theorem 8, $\mathrm{O}(s_i) = \mathrm{I}^{-1}(d_i) = 0$ is associated with the $(n\text{-}1)$-stage CP$^{\text{min}}$ type MIN $\mathbf{J}_{00}$ and $\mathrm{O}(s_i) = \mathrm{I}^{-1}(d_i) = 1$, with the $(n\text{-}1)$-stage CP$^{\text{min}}$ type MIN $\mathbf{J}_{11}$. Similarly, for the 1-state case, any input S with bit $\mathrm{O}(s_i) = 0$ (1) can communicate with any

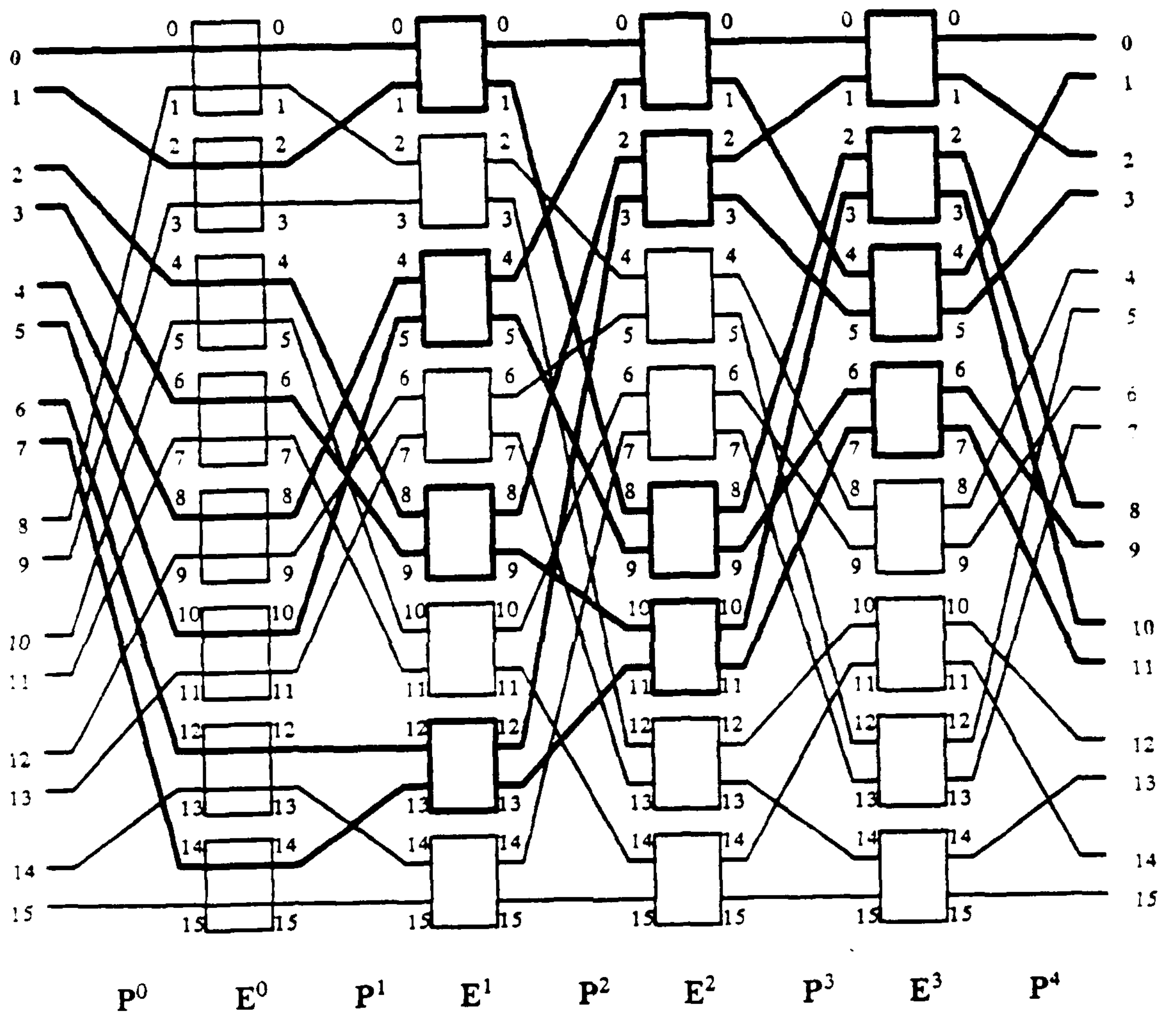


Fig. 2.9. The 3-stage subnetwork J_{00} on the MIN shown in Fig. 2.5 is formed by forcing all the switching elements at stage 0.

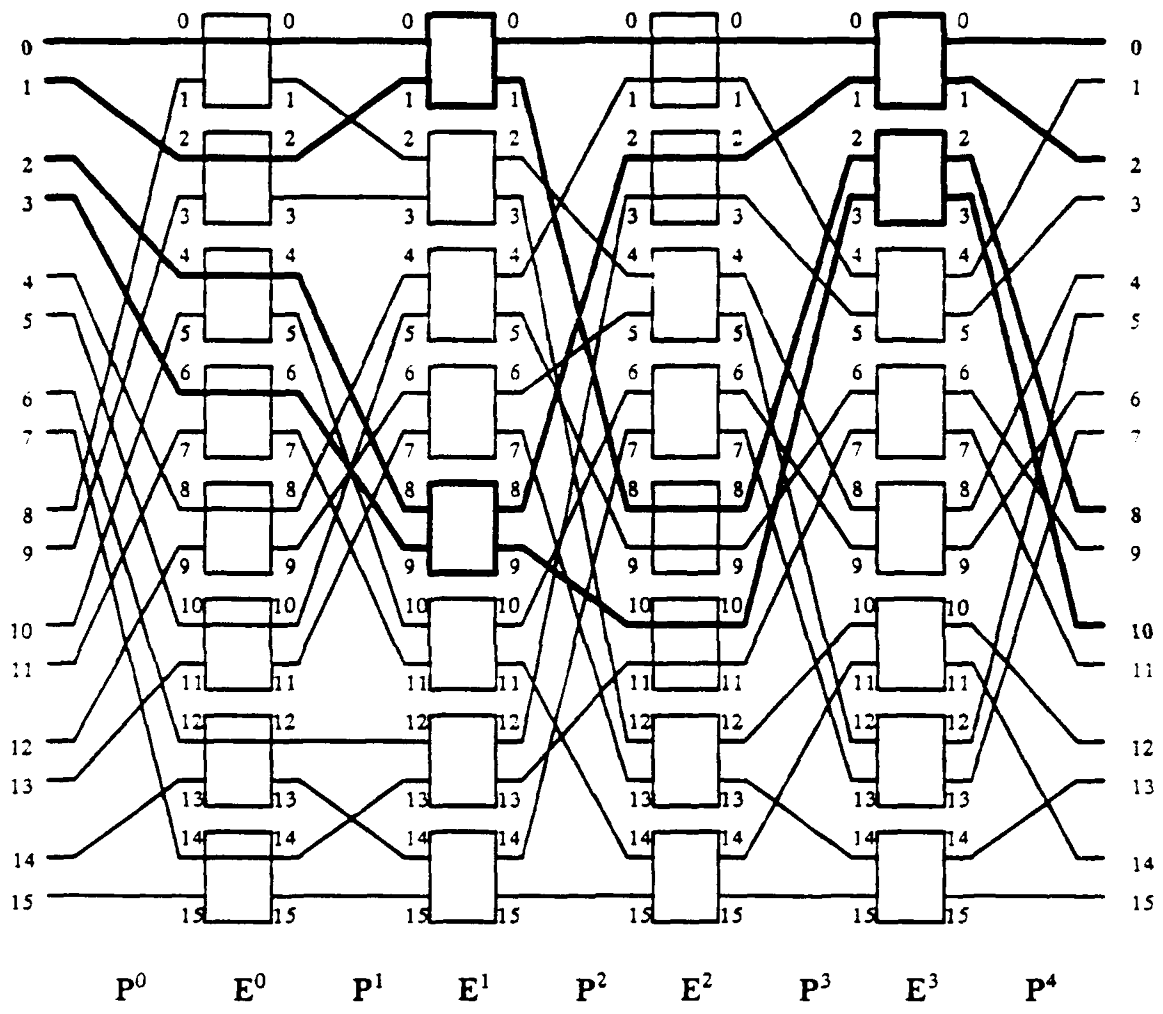


Fig. 2.10. One of the two 2-stage subnetworks on J_{00} shown in Fig. 2.5. is formed by forcing all the switching elements at stage 2.

output D with bit $\mathbf{I}^{-1}(d_i) = 1$ (0) such that the joined (n-1)-stage CP^{min} type MIN $\mathbf{J}_{01}$ ($\mathbf{J}_{10}$) is formed. □

For example, by forcing all the switching elements at stage 0 on the MIN shown in Fig. 5 to 0-state, we have one of the two 8×8 3-stage subnetworks, $\mathbf{J}_{00}$, shown boldly in Fig. 2.9. For subnetwork $\mathbf{J}_{00}$, all the input addresses agree in bit $\mathbf{O}(s_0) = s_3 = 0$ and all the output addresses agree in bit $\mathbf{I}^{-1}(d_0) = d_2 = 0$. The general form of transition sequences on $\mathbf{J}_{00}$ is as follows:

Similarly, by forcing all the switching elements at stage 2 on J00, we have one of the two 4×4 2-stage subnetworks shown in Fig. 2.10. All the input addresses agree in bit $\mathbf{O}(s_0) = s_3$ and $\mathbf{O}(s_2) = s_2$; all the output addresses agree in bit $\mathbf{I}^{-1}(d_0) = d_2$ and $\mathbf{I}^{-1}(d_2) = d_0$. Note that not every Banyan type MIN can be partitioned. For example, the MIN in Fig. 6(b) is not partitionable.

2.5. SUMMARY

In this chapter, we propose a class of Banyan type MINs defined by CP(n) type connection patterns, denoted as $\log_2 N$ CP^{min}. This class includes all the famous MINs presented previously in the literature as special cases. We show that the topological structure of each network in this class can be specified by two characteristic functions and particularly the topological equivalence among networks can be interpreted as linear transformations on characteristic functions. Based on characteristic functions, their topology-related functional behavior, such as the simple bit-directed routing scheme, has been discussed. Actually, our methodology can easily be extended to all Banyan type MINs, where, in general, their characteristic functions are two-dimensional matrices rather than one-dimensional permutation functions. The proposed approach also provides a good description for the closed form of all the passable permutations on various MINs, the condition for conflict-free multiple paths and the network partitioning.

CHAPTER 3

TRANSFORMATION RULES FOR MULTISTAGE INTERCONNECTION NETWORKS

3.1. INTRODUCTION

The general concept of parallel supercomputers is to employ a communication network to interconnect a large number of processors and a large number of memory modules in a way that processors can communicate to others and memory modules can be simultaneously accessed without conflict. Although the crossbar network does provide such a capability, it is not economically practical when the number of processors and modules becomes large. A realistic alternative is to use multistage interconnection networks (MINs). Typically, they are designed using $\log_2 N$ stages of $\frac{N}{2}$ 2×2 switching elements and $(\log_2 N+1)$ fixed connection patterns to connect N inputs to N outputs such that only the minimum number of switching elements are required to provide full access capability from all the inputs to all the outputs and a unique communication path from any input to any output. Examples are the Omega network [Law75], Baseline and Reverse Baseline network [WuFe80], Indirect Binary Cube network [Pea77], Delta network [Pat81], Cube network [SiSm78], etc. Since the capability of each network in terms of non-conflict permutation communications are different, different networks can be selected to efficiently support different application needs. For example, in [Law75], the Omega network is selected for supporting matrix computations on an array processor. An important question arises as to whether we can reuse the algorithms/software (which have been developed on a system using a MIN) on another system which employs a different MIN. It is well known that as long as two networks are equivalent, there exists, theoretically, a one-to-one mapping function between them such that by relabeling the processors and memories and loading data into the memories based on the new labels, one network can simulate the other. For example, in [WuFe80], it is shown that several networks are functionally equivalent to the Baseline network. So far, many theoretical studies have been performed. For example, in [Agr83], it is shown that networks with full connection and strict buddy property are functionally equivalent; in [BeFo88], necessary condition for networks to be equivalent are also established. In general, their studies are still at a very abstract level such that transformation rules between networks are still not available. In fact, it is extremely difficult (if not impossible) to derive those transformation rules from those theoretical studies. Therefore, all the research works in the area of network equivalence are still very far away from the practical applications. In this chapter, we will present the mapping functions between equivalent networks in a concise way such that, for the first time, the equivalence can be fully utilized in the design of supercomputers.

3.2. BIT-PERMUTE-COMPLEMENT MULTISTAGE INTERCONNECTION NETWORKS

In this section, a rather general class of equivalent networks based on bit-permute-complement shuffles is outlined, which includes all the above mentioned networks as special cases and requires a very simple destination address based routing scheme. The connection patterns of this class of networks are defined by the family of *Bit−Permute−Complement* (BPC) type permutations, denoted by $\{\beta_n\}$ and defined as follows.

DEFINITION 1: Let $l = (l_{n-1}, \ldots, l_1, l_0)$ be a number in $\{0, 1, \ldots, N-1\}$. A permutation $\beta \in \{\beta_n\}$ is specified by an n-tuple vector $\hat{\beta} = (\lambda_{n-1}\theta(n-1), \ldots, \lambda_1\theta(1), \lambda_0\theta(0))$, where $(\theta(n-1), \ldots, \theta(1), \theta(0))$ is a permutation of $(n-1, \ldots, 1, 0)$ and $\lambda_i \in \{-1, 1\}$, $0 \le i \le n-1$, such that $\beta(l_{n-1}, \ldots, l_1, l_0) = (m_{n-1}, \ldots, m_1, m_0)$, where $m_i = l_{\theta(i)}$ if $\lambda_i = 1$, else $m_i = 1 - l_{\theta(i)}$ if $\lambda_i = -1$. □

In other words, $\beta(l)$ is obtained from l by first permuting the bits of the binary representation of l and then complementing a subset of bits according to the vector V. For example, the *bit−reversal* permutation ρ is one of the BPC type permutations and $\rho(l) = (l_0, l_1, \ldots, l_{n-2}, l_{n-1})$, where $\hat{\rho} = (0, 1, \ldots, n-1)$. Another example is the perfect shuffle permutation ζ with $\zeta(l) = (l_{n-2}, \ldots, l_0, l_{n-1})$, where $\hat{\zeta} = (n-2, \cdots, 0, n-1)$. Any connection pattern defined by a permutation β has a link connecting lth output port of one switching stage to $\beta(l)$th input port of the next switching stage, for all $0 \le l \le 2^n - 1$. Similarly, the inverse function β^{-1} is defined as follows.

DEFINITION 2: Let β^{-1} be the inverse function of $\beta \in \{\beta_n\}$. The function $\beta^{-1} \in \{\beta_n\}$ is specified by an n-tuple vector $\hat{\beta}^{-1} = (\lambda_{\theta^{-1}(n-1)}\theta^{-1}(n-1), \ldots, \lambda_{\theta^{-1}(1)}\theta^{-1}(1), \lambda_{\theta^{-1}(0)}\theta^{-1}(0))$

where θ^{-1} is the inverse function of θ. □

Let α and β be two permutation functions in $\{\beta_n\}$ and specified by vectors $\hat{\alpha}$ and $\hat{\beta}$, respectively. The composition of α and β is denoted as $\alpha \cdot \beta$ such that $\alpha \cdot \beta(l) = \beta(\alpha(l))$ (i.e., the composed functions are performed from left to right) and is specified by a vector $\hat{\alpha} \cdot \hat{\beta}$. In the following, the notation $\alpha \cdot \beta$ and $\alpha\beta$ will be used interchangeably. For example, consider $n = 4$ and let β be specified by $\hat{\beta} = (-2,-1,0,3)$ and $l = (1,0,0,1)$. We have $m_3 = 1 - l_2$, $m_2 = 1 - l_1$, $m_1 = l_0$ and $m_0 = l_3$. Hence, $\beta(l) = (1,1,1,1)$. Similarly, it is easy to obtain that $\hat{\beta}^{-1} = (0,-3,-2,1)$ and $\beta^{-1}(l) = (l_0, 1 - l_3, 1 - l_2, l_1) = (1,0,1,0)$. If α is specified by $\hat{\alpha} = (-1,2,3,-0)$, then $\alpha\beta(l) = \beta(\alpha(l)) = \beta(1,0,1,0) = (1,0,0,1)$ and the composition $\alpha\beta$ is specified by the vector $\hat{\alpha} \cdot \hat{\beta} = (-1,2,3,-0) \cdot (-2,-1,0,3) = (-2,-3,-0,-1)$.

It is clear that an $N \times N$ (i.e., N inputs and N outputs) MIN (see Fig. 3.1.) can be represented by a sequence of BPC permutation and exchange operations

$$P^0 . E^0 . P^1 \ . \ . \ . \ P^{(n-1)} . E^{(n-1)} . P^n$$

where $n = \log_2 N$, each P^i $(0 \le i \le n)$ represents a BPC permutation operation, and each E^i $(0 \le i \le n-1)$ denotes an exchange operation. That is, the ith connection pattern corresponds to the BPC permutation operation P^i and the ith switching stage corresponds to the exchange operation E^i. Each E^i represents a BPC permutation operation such that the least significant bit (LSB) of its operand is replaced by another bit which is either the original bit or its complement, i.e., E^i is specified by a vector either $(n-1, n-2, \ldots, 1, 0)$ or $(n-1, n-2, \ldots, 1, -0)$. In other words, $E^i(l_{n-1}, \ldots, l_1, l_0) = (l_{n-1}, \ldots, l_1, l_0)$ or $(l_{n-1}, \ldots, l_1, \bar{l}_0)$. Assume that the addresses of both network inputs and outputs are numbered from 0 to $N-1$ following a natural order in the drawing. (Note that each address can be represented by an n-bit binary number.) Consider a communication connection to be established from a network input with address S

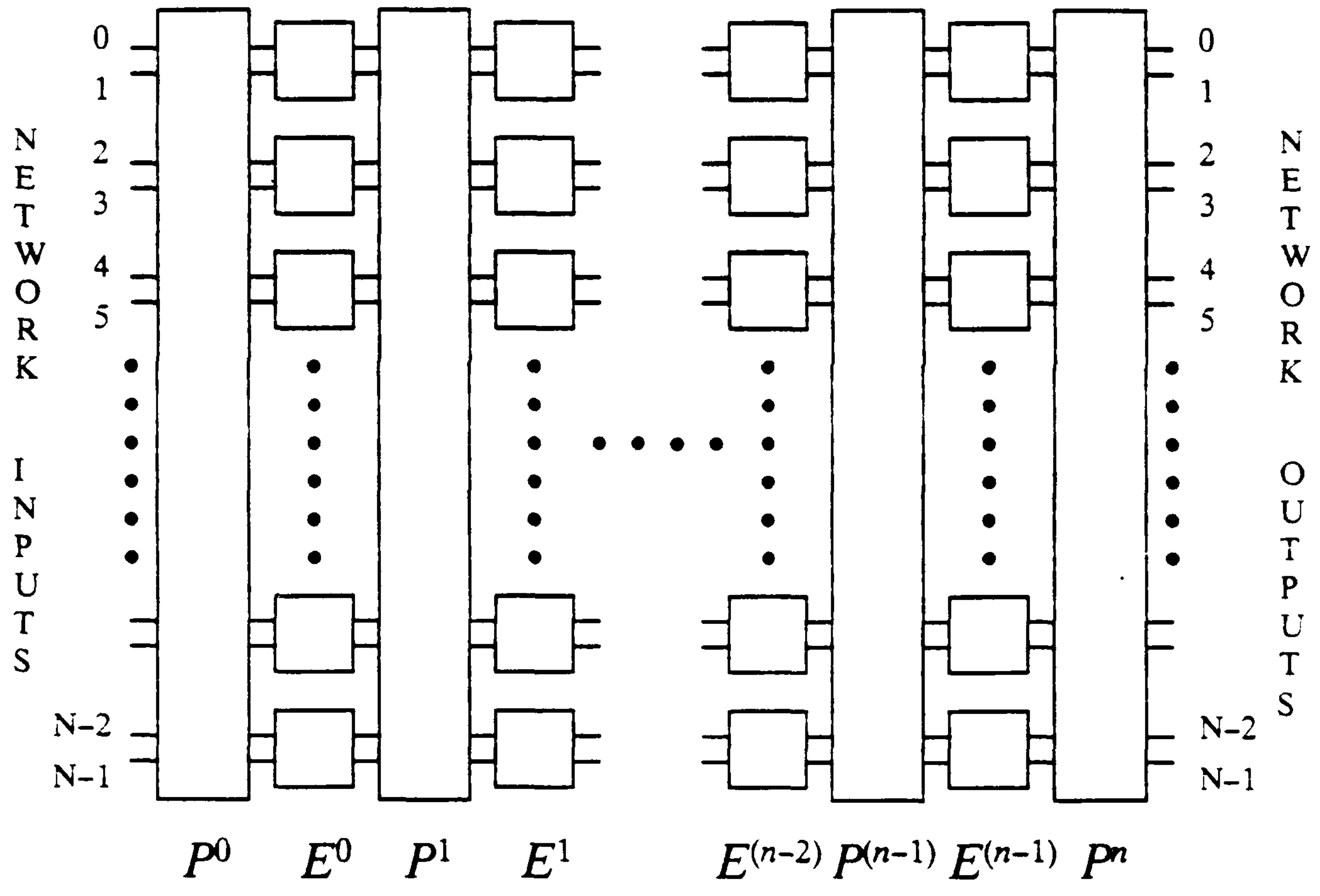


Fig. 3.1. An $N \times N$ MIN is represented by a sequence of BPC permutation and exchange operations.

$= (s_{n-1}, \ldots, s_1, s_0)$ to a network output with address $D = (d_{n-1}, \ldots, d_1, d_0)$. During the propagation, the source address bits (i.e., s_i's or $\overline{s_i}$'s, the complement of s_i's) are replaced by destination bits (i.e., d_i's or $\overline{d_i}$'s) one by one at each switching stage. The orders for source address bits to be removed and for destination bits to be introduced are determined by the BPC permutation operations (i.e., P^i's). In principle, during each BPC permutation operation, a source address bit (or its complement) is moved to the position of LSB and, at the next switching stage of exchange operation, is replaced by a destination bit (or its complement) at the same bit location.

For example, the BPC permutation operations used in the Omega network are perfect shuffle connections (which are left rotation operations). Any path connecting a network input $S = (s_{n-1}, \ldots, s_1, s_0)$ to a network output $D = (d_{n-1}, \ldots, d_1, d_0)$ on an Omega network, can be uniquely expressed by the following transition sequence:

$$S = (s_{n-1}, s_{n-2}, \ldots, s_1, s_0)$$

$$S^0 = (s_{n-2}, s_{n-3}, \ldots, s_0, s_{n-1})$$

$$D^0 = (s_{n-2}, s_{n-3}, \ldots, s_0, d_{n-1})$$

$$S^1 = (s_{n-3}, s_{n-4}, \ldots, s_0, d_{n-1}, s_{n-2})$$

$$D^1 = (s_{n-3}, s_{n-4}, \ldots, s_0, d_{n-1}, d_{n-2})$$

......

$$S^i = (s_{n-2-i}, s_{n-3-i}, \ldots, s_0, d_{n-1}, \ldots, d_{n-i}, s_{n-1-i})$$

$$D^i = (s_{n-2-i}, s_{n-3-i}, \ldots, s_0, d_{n-1}, \ldots, d_{n-i}, d_{n-1-i})$$

......

$$S^{n-1} = (d_{n-1}, d_{n-2}, \ . \ . \ . \ , d_1, s_0)$$

$$D^{n-1} = (d_{n-1}, d_{n-2}, \ . \ . \ . \ , d_1, d_0)$$

$$S^n = (d_{n-1}, d_{n-2}, \ . \ . \ . \ , d_1, d_0)$$

$$= D \, .$$

In the transition sequence, each number S^i, $0 \leq i \leq n - 1$, is the address of the input port through which the path traverses stage i and each number D^i, $0 \leq i \leq n - 1$, the output port through which the path traverses stage i. Obviously, either $[S^i]_{n-1:1}$ or $[D^i]_{n-1:1}$ (i.e., the first n-1 bits of the binary representation of S^i or D^i) is the address of the switching element through which the path traverses stage i. Obviously, source bit $s_{(n-1)-i}$ is moved to the LSB position of S^i at stage i and is replaced with the destination bit $d_{(n-1)-i}$ of D in the LSB position of D^i. Note that in the transition sequence we have the following relations: $P^0(S) = S^0$, $E^i(S^i) = D^i$ for all $0 \leq i \leq n-1$, and $P^i(D^{i-1}) = S^i$ for all $1 \leq i \leq n$. Therefore, the order for the source bits to be removed is $n-1, n-2, n-3, \ . \ . \ . \ , 2, 1, 0$ and the order for the destination bits to be introduced is also $n-1, n-2, n-3, \ . \ . \ . \ , 2, 1, 0$. Here, two vectors $\hat{O}$ = $(0,1,2 \ ,...., \ n-2, n-1)$ and $\hat{I}^{-1}$ = $(0,1,2 \ ,...., \ n-2, n-1)$ which correspond to two permutation functions O and I^{-1} in $\{\beta_n\}$, are used to represent these two sequences, respectively. Note that by using

$$\hat{O} = (\ \hat{o}(n-1), \hat{o}(n-2) \ , \ . \ . \ . \ , \ \hat{o}(2), \hat{o}(1), \hat{o}(0) \ )$$

and

$$\hat{I}^{-1} = (\ \hat{i}^{-1}(n-1), \hat{i}^{-1}(n-2) \ , \ . \ . \ . \ , \ \hat{i}^{-1}(2), \hat{i}^{-1}(1), \hat{i}^{-1}(0) \ )$$

we mean that at stage j source bit $s_{\hat{o}(j)}$ will be removed and replaced by destination bit $d_{\hat{i}^{-1}(j)}$. Moreover, the meaning of the permutation function I is as follows: if bit $s_{\hat{o}(j)}$ is replaced by bit d_j at stage j, then the order of bits of $I(D)$ represents the disturbed order of bits of D. For the Omega network, we have $I^{-1}(D) = I(D)$. Similarly, the $\hat{O}$ and $\hat{I}^{-1}$ for the Baseline network [WuFe80] are given as follows:

$$\hat{O} = (n-1, n-2, \ldots, 1, 0)$$

$$\hat{I}^{-1} = (0, 1, \ldots, n-2, n-1).$$

It is clear that there are a huge number of MINs in this class which are constructed by the BPC permutation connections and possess the unique-path and full-access properties. Here we use the term *bit−permute−complement* MINs to represent this class. Generally speaking, the bit-permute-complement MINs are a class of topologically equivalent networks which have the similar routing behavior and thus, the similar expression of transition sequences like Omega networks. This class of MINs includes the six networks mentioned in [WuFe80] as special cases. The connection patterns used between stages of them are a specified set from $\{\beta_n\}$. Their transition sequence which represents any path connecting network input $S = (s_{n-1}, \ldots, s_1, s_0)$ to network output $D = (d_{n-1}, \ldots, d_1, d_0)$ has the following properties.

(1) Each bit of the sources S (or its complement) will be permuted to the position of LSB in some S^i and then be replaced by a bit of the destination D (or its complement) in D^i, $0 \le i \le n - 1$. Therefore, there exist two permutation functions O , $I^{-1} \in \{\beta_n\}$ such that $O(S)$ corresponds to the order for bits of S to be permuted to the position of LSB (i.e, $[O(S)]_i$ is the LSB in S^i) and $I^{-1}(D)$ corresponds to the order for bits of D to replace bits of S (i.e., $[I^{-1}(D)]_i$ replaces $[O(S)]_i$ of D^i). The physical meaning of permutation function I is as follows: if the ith bit of a number $X = (x_{n-1}, x_{n-2}, \ldots, x_1, x_0)$ instead of bit $[I^{-1}(D)]_i$ replaces $[O(S)]_i$ in D^i, then $I(X)$ represents the final destination where the source S will reach.

(2) The data is routed from input port $[O(S)]_i$ to output port $[I^{-1}(D)]_i$ of the switching element at stage i and the address of the switching element is either $[S^i]_{n-1:1}$ or $[D^i]_{n-1:1}$.

(3) The routing scheme of this class of MINs can be described as follows. Let the symbol @ represent the exclusive-or operation. Bit $[I^{-1}(D)]_i$ is used as the routing tag for the switching element at stage i such that the data is routed from input port $[O(S)]_i$ to output port $[I^{-1}(D)]_i$. Bit $[O(S)]_i$ @ $[I^{-1}(D)]_i$ is used to determine the state of the switching element at stage i if global routing is considered (i.e., if $[O(S)]_i$ @ $[I^{-1}(D)]_i = 0$ then the switching element will be in a straight connection state (i.e., 0 state), else the switching element will be in a exchange connection (i.e., 1 state)). After the path traverses stage i, bit $[O(S)]_i$ (i.e., the label of the input port from which the incoming data comes) is preserved to recover the information of the source address. We call this kind of routing scheme as the *source−preserved* and *destination−oriented* routing scheme. Note that not all the MINs with full access capability and unique-path property possess this kind of simple routing scheme.

Functions O and I are referred to as *characteristic* functions. It can be shown that it is sufficient to characterize the structure and routing behavior of any MIN in the class of bit-permute-complement MINs by using these two functions. Note that given any two permutation functions O and I in $\{\beta_n\}$, there are many different corresponding sequences of BPC permutation operations $(P^0, P^1, ..., P^{n-1}, P^n)$ such that the MIN constructed by any one of them can be characterized by functions O and I. Each corresponding sequence of BPC permutation operations represents a drawing of a MIN. It can be proved that there are totally $[2^{n-1} \cdot (n-1)!]^n$ different drawings (i.e., MINs) specified by the same pair of characteristic functions. It can also be shown that the permutation capability of any MIN in this class is uniquely characterized by these two functions. Thus, we say two MINs in the class of bit-permute-complement MINs are *functionally equivalent* if their characteristic functions are the same.

For example, in Fig. 3.2, a 16×16 bit-permute-complement MIN defined by a sequence of BPC permutation operations is shown. Let the characteristic functions of this network, O, I and I^{-1}, be specified by vectors $\hat{O}$, $\hat{I}$ and $\hat{I}^{-1}$, respectively. Let the connection pattern P^i, $0 \le i \le n$, be specified by vector $\hat{P}^i$. We have $\hat{P}^0$ = (2,-1,-0,3), $\hat{P}^1$ = (2,3,0,-1), $\hat{P}^2$ = (-0,1,3,-2), $\hat{P}^3$ = (2,-0,3,1), $\hat{P}^4$ = (1,3,0,2), $\hat{O}$ = (-1,-2,0,3), $\hat{I}$ = (-1,0,3,-2), and $\hat{I}^{-1}$ = (1,-0,-3,2). For any path connecting a source S to a destination D, bit d_2 is used as the routing tag at stage 0, bit $\bar{d}_3$ is used as the routing tag at stage 1, bit $\bar{d}_0$ is used as the routing tag at stage 2, and bit d_1 is used as the routing tag at stage 3. The states of switching elements from stage 0 to stage 3 are determined by $s_3 @ d_2$, $s_0 @ \bar{d}_3$, $\bar{s}_2 @ \bar{d}_0$, and $\bar{s}_1 @ d_1$, respectively. Hence, for the path connecting $S = 1$ to $D = 12$, the routing tags are $(d_1, \bar{d}_0, \bar{d}_3, d_2)$ = (0,1,0,1) and states are $(\bar{s}_1 @ d_1, \bar{s}_2 @ \bar{d}_0, s_0 @ \bar{d}_3, s_3 @ d_2)$ = (1,0,1,1).

3.3. NETWORK TRANSFORMATION RULES

The transformation rules in order to transform a MIN (which is characterized by two functions O_1 and I_1 and denoted as MIN_1) into another MIN (which is characterized by another two functions O_2 and I_2 and denoted as MIN_2) is discussed in this section.

THEOREM 1: By adding two fixed connection patterns α and β at the input and output sides, respectively, of a MIN (which is characterized by two functions O_1 and I_1 and denoted as MIN_1) which are defined as follows:

$$\alpha = O_2 \cdot O_1^{-1}$$

$$\beta = I_1^{-1} \cdot I_2 ,$$

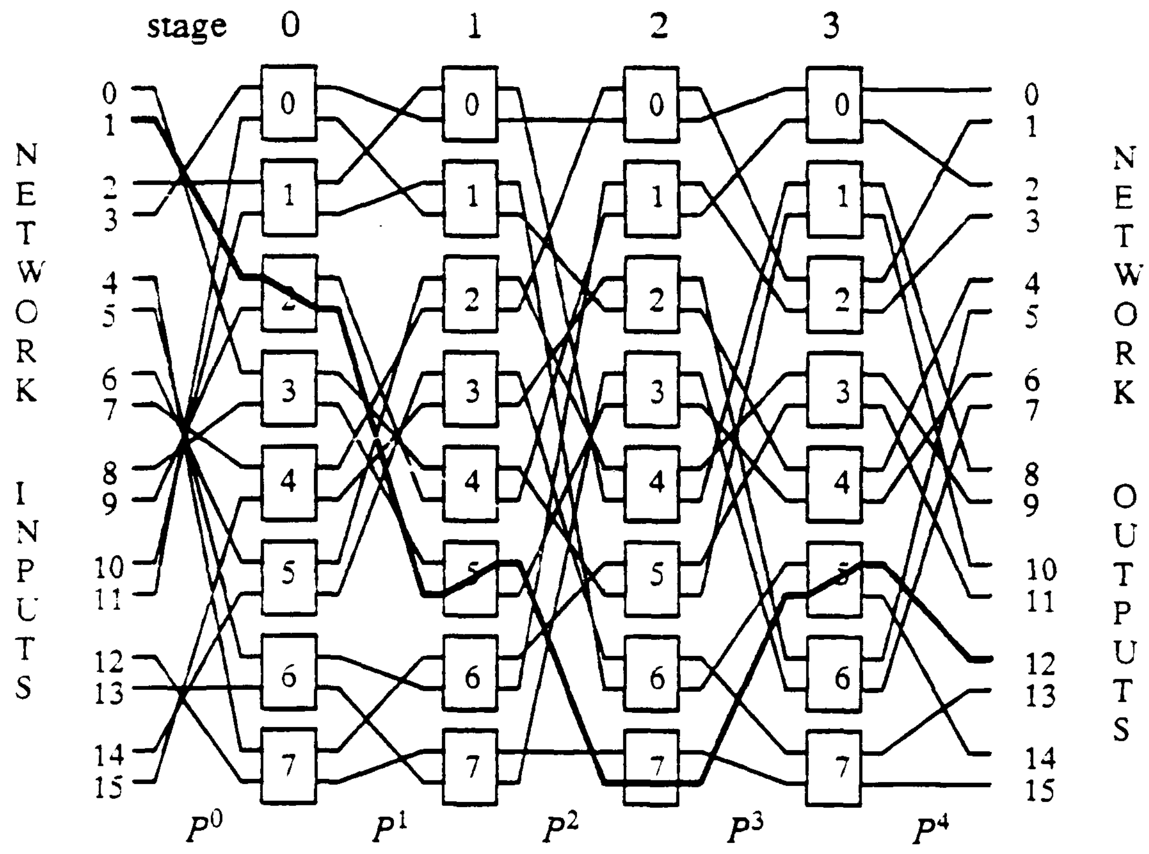


Fig. 3.2. A 16×16 bit-permute-complement MIN defined by a sequence of BPC permutation operations.

then the resultant MIN becomes a MIN_2 characterized by another two functions O_2 and I_2.

PROOF: (see Fig. 3.3) Two fixed connection patterns α and β are added to the input and output sides of MIN_1. Let us call it the new MIN_1. As discussed in previous sections, for the transition sequence representing the path connecting a source S to a destination D in the new MIN_1, $\alpha \cdot O_1(S)$ corresponds to the order for bits of S to be permuted to the position of LSB (i.e, $[\alpha \cdot O_1(S)]_i$ is the LSB in S^i) and $I_1 \cdot \beta(D)$ corresponds to the disturbed order for bits of D, if we use bit d_i to replace $[\alpha \cdot O_1(S)]_i$ in D^i. Obviously, the new MIN_1 can be characterized by two permutation functions: $\alpha \cdot O_1$ and $I_1 \cdot \beta$. Since $\alpha = O_2 \cdot O_1^{-1}$ and $\beta = I_1^{-1} \cdot I_2$, we have $\alpha \cdot O_1 = O_2 \cdot O_1^{-1} \cdot O_1 = O_2$ and $I_1 \cdot \beta = I_1 \cdot I_1^{-1} \cdot I_2 = O_2$. That is, the new MIN_1 is equivalent to MIN_2. $\square$

A renumbering scheme instead of using connection patterns can transform a MIN to another. Thus, another way to describe Theorem 1 is as follow:

THEOREM 2: By renumbering the addresses of network inputs and outputs of a MIN (which is characterized by two functions O_1 and I_1 and denoted as MIN_1) in the following way:

$$\text{the new address of network input } S = O_1 \cdot O_2^{-1}(S)$$

$$\text{the new address of network output } D = I_1^{-1} \cdot I_2(D),$$

then the resultant MIN becomes a MIN_2 which is characterized by another two functions O_2 and I_2.

PROOF: It is clear that if we renumber the address of network input S with the new address $\alpha^{-1}(S) = O_1 \cdot O_2^{-1}(S)$ and the address of network output D with the new address $\beta(D) = I_1^{-1} \cdot I_2(D)$, then it is equivalent to adding two connection patterns α and β at the

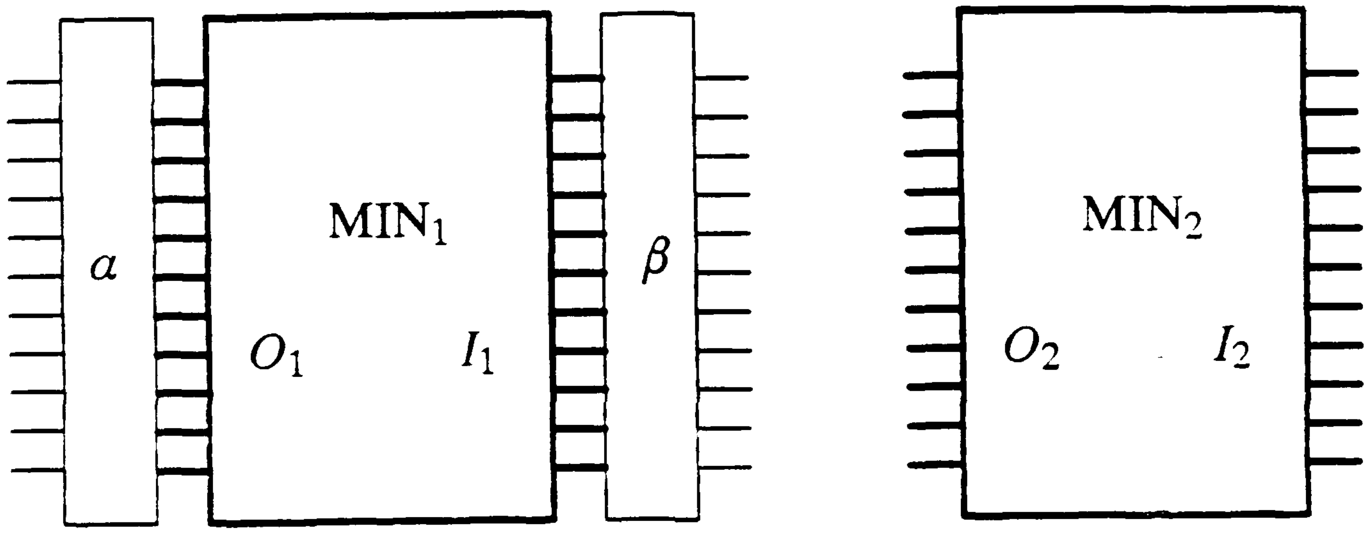


Fig. 3.3. Transformation between MINs.

input and output sides of MIN_1. Thus, as mentioned in Theorem 1, the resultant MIN becomes a MIN_2 which is characterized by two functions O_2 and I_2. $\square$

The new MIN_1 generated by applying Theorem 1 or Theorem 2 is functionally equivalent to MIN_2 except drawing. Both MIN_1 and MIN_2 follow the same routing scheme, i.e., they use the same routing tag $I_2^{-1}(D)$ for connecting the source S to the destination D and bit $[O_2(S)]_i$ @ $[I_2^{-1}(D)]_i$ to control the state of the switching element at stage i if global routing is considered. For example, consider the case where MIN_1 is a 16×16 Omega network and MIN_2 is the 16×16 bit-permute-complement MIN shown in Fig. 3.2. As mentioned above, their characteristic functions O_1, I_1, O_2, and I_2 can be specified by the following vectors: $\hat{O}_1 = (0,1,2,3)$, $\hat{O}_1^{-1} = (0,1,2,3)$, $\hat{I}_1 = (0,1,2,3)$, $\hat{I}_1^{-1} = (0,1,2,3)$, $\hat{O}_2 = (-1,-2,0,3)$, $\hat{O}_2^{-1} = (0,-2,-3,1)$, $\hat{I}_2 = (-1,0,3,-2)$, and $\hat{I}_2^{-1} = (1,-0,-3,2)$. From Theorem 1, if we add two connection patterns α and β at input and output sides of MIN_1, such that

$$\hat{\alpha} = \hat{O}_2 \cdot \hat{O}_1^{-1} = (-1,-2,0,3) \cdot (0,1,2,3)^{-1} = (-1,-2,0,3) \cdot (0,1,2,3) = (3,0,-2,-1)$$

$$\hat{\beta} = \hat{I}_1^{-1} \cdot \hat{I}_2 = (0,1,2,3)^{-1} \cdot (-1,-2,0,3) = (0,1,2,3) \cdot (-1,-2,0,3) = (-2,3,0,-1),$$

then the resultant MIN_1 which is functionally equivalent (except that the arrangement of positions of switching elements is different) to MIN_2 is shown in Fig. 3.4.

However, if we apply Theorem 2, then the network inputs and outputs are renumbered according to functions α^{-1} and β, respectively. That is,

$$\hat{\alpha}^{-1} = \hat{O}_1 \cdot \hat{O}_2^{-1} = (0,1,2,3) \cdot (-1,-2,0,3)^{-1} = (0,1,2,3) \cdot (0,-2,-3,1) = (3,-1,-0,2)$$

$$\hat{\beta} = (-2,3,0,-1)$$

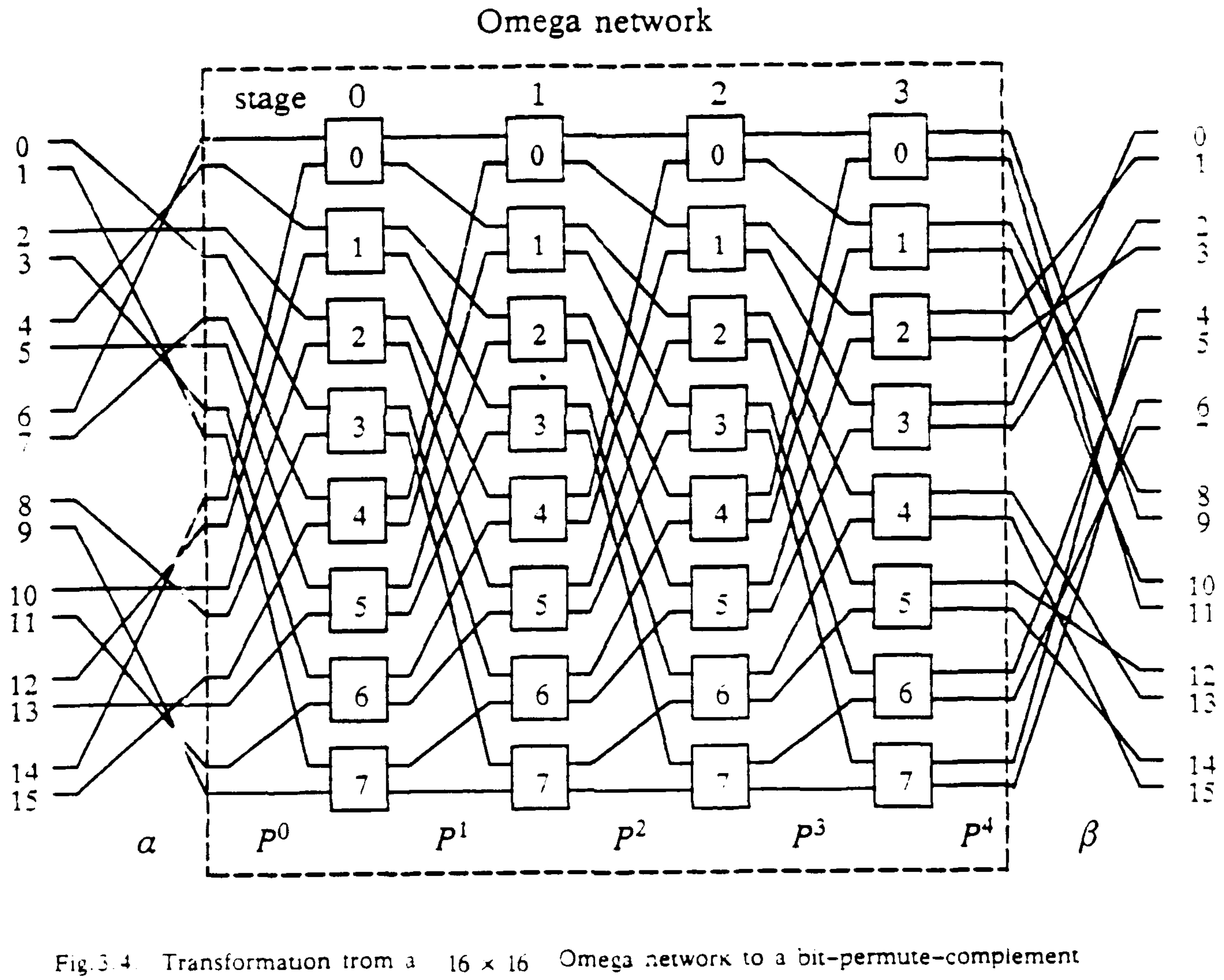


Fig.3.4 Transformation from a 16 × 16 Omega network to a bit-permute-complement MIN shown in Fig. 3.2.

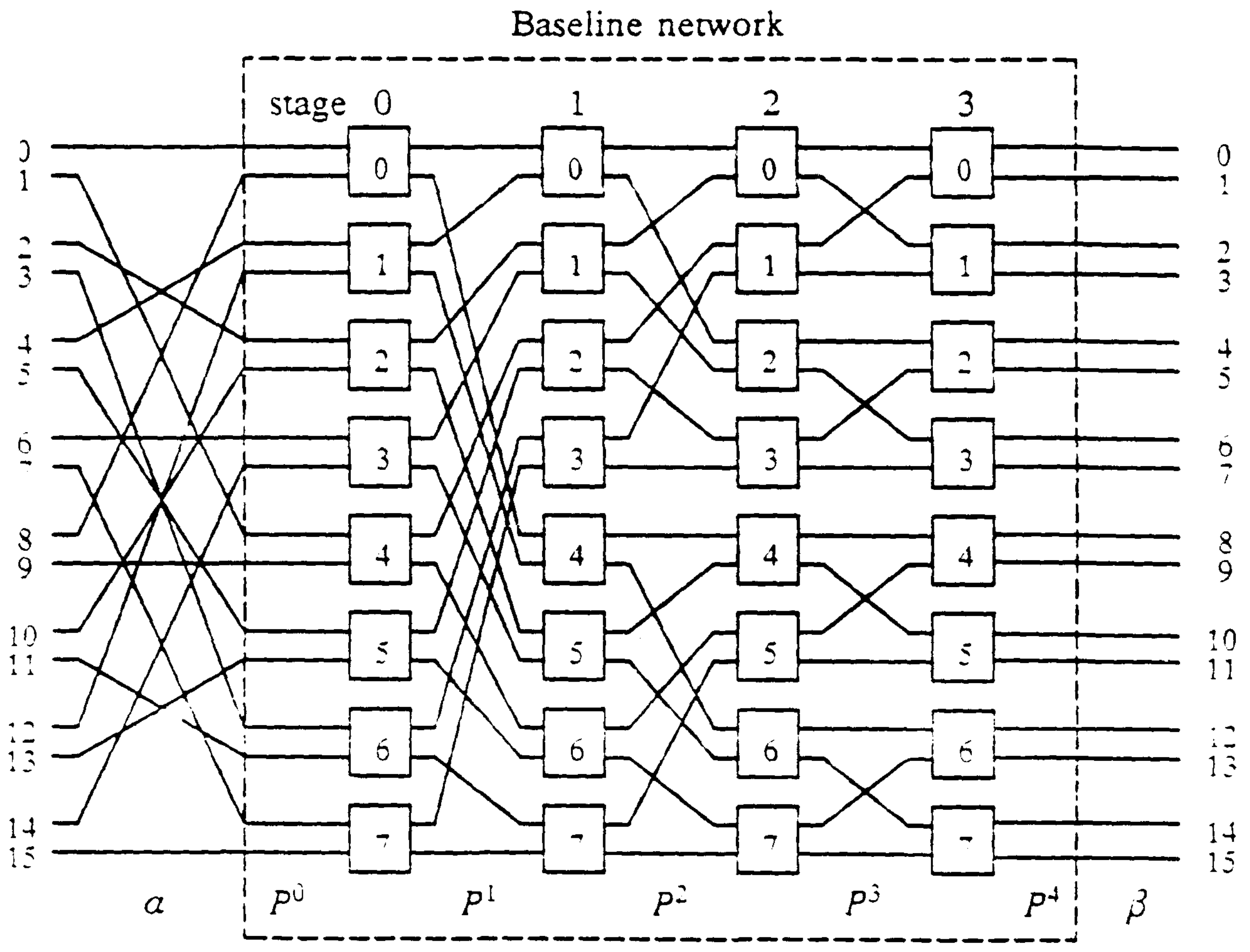


Fig. 3.5. Transformation from a 16 × 16 Baseline network to a 16 × 16 Omega network.

Thus, the new address of network input $S = (s_3,s_2,s_1,s_0)$ is $\alpha^{-1}(S) = (s_3,\bar{s}_1,\bar{s}_0,s_2)$ and the new address of network output $D = (d_3,d_2,d_1,d_0)$ is $\beta(D) = (\bar{d}_2,d_3,d_0,\bar{d}_1)$.

Consider another example where MIN_1 is a 16×16 Baseline network and MIN_2 is a 16×16 Omega network. Again, we can derive the characteristic functions O_1, I_1, O_2, and I_2 as follows: $\hat{O}_1 = (3,2,1,0)$, $\hat{O}_1^{-1} = (3,2,1,0)$, $\hat{I}_1 = (0,1,2,3)$, $\hat{I}_1^{-1} = (0,1,2,3)$, $\hat{O}_2 = (0,1,2,3)$, $\hat{O}_2^{-1} = (0,1,2,3)$, $\hat{I}_2 = (0,1,2,3)$, and $\hat{I}_2^{-1} = (0,1,2,3)$. From Theorem 1, if we add two connection patterns α and β at input and output sides of MIN_1, such that

$$\hat{\alpha} = \hat{O}_2 \cdot \hat{O}_1^{-1} = (0,1,2,3) \cdot (3,2,1,0)^{-1} = (0,1,2,3) \cdot (3,2,1,0) = (0,1,2,3)$$

$$\hat{\beta} = \hat{I}_1^{-1} \cdot \hat{I}_2 = (0,1,2,3)^{-1} \cdot (0,1,2,3) = (0,1,2,3) \cdot (0,1,2,3) = (3,2,1,0),$$

then the resultant MIN which is functionally equivalent (except that the arrangement of positions of switching elements is different) to MIN_2 is shown in Fig. 3.5.

However, if we apply Theorem 2, then the network inputs and outputs are renumbered according to functions α^{-1} and β, respectively. That is,

$$\hat{\alpha}^{-1} = \hat{O}_1 \cdot \hat{O}_2^{-1} = (3,2,1,0) \cdot (0,1,2,3)^{-1} = (3,2,1,0) \cdot (0,1,2,3) = (0,1,2,3)$$

$$\hat{\beta} = (3,2,1,0)$$

Thus, the new address of network input $S = (s_3,s_2,s_1,s_0)$ is $\alpha^{-1}(S) = (s_0,s_1,s_2,s_3)$ and the new address of network output $D = (d_3,d_2,d_1,d_0)$ is $\beta(D) = (d_3,d_2,d_1,d_0)$.

The next theorem describes the relative positions of each switching element before and after a network transformation. Consider the case where MIN_1 is transformed to MIN_2. The method to find the relative positions is based on the criterion that two paths connecting the same source and the same destination of both MIN_1 and MIN_2 pass the same switching element at each stage. Thus, if the address of each switching element of MIN_1 is replaced by its relative address in MIN_2 or if the address of each switching element of MIN_2 is replaced by its relative address in MIN_1, then both MIN_1 and MIN_2 become the same drawing. Let MIN_1 be defined by a sequence of BPC operations $(\Phi_0, \Phi_1, ..., \Phi_{n-1}, \Phi_n)$ and MIN_2, $(\Psi_0, \Psi_1, ..., \Psi_{n-1}, \Psi_n)$, for all $\Phi_i, \Psi_i \in \{\beta_n\}$. Thus, the transformed MIN_1 is defined by the sequence of BPC operations $(\alpha{\cdot}\Phi_0, \Phi_1, ..., \Phi_{n-1}, \Phi_n{\cdot}\beta)$.

THEOREM 3: By applying Theorem 1 or Theorem 2, consider the case where MIN_1 is transformed to MIN_2. Let $SW_1[i, j]$ represent the jth switching element at stage i of the transformed MIN_1 and $SW_2[i^*, j^*]$ represent the j^*th switching element at stage i^* of MIN_2 where $0 \le i, i^* \le n-1$ and $0 \le j, j^* \le N/2-1$. The following relation transforms both MIN_1 and MIN_2 to the same drawing:

$$i^* = i,$$

$$j^* = [\Phi_i^{-1}{\cdot}\Phi_{i-1}^{-1} \cdots \Phi_0^{-1}{\cdot}\alpha^{-1}{\cdot}\Psi_0{\cdot}\Psi_1 \cdots \Psi_i(2j)]_{n-1:1},$$

or

$$j = [\Psi_i^{-1}{\cdot}\Psi_{i-1}^{-1} \cdots \Psi_0^{-1}{\cdot}\alpha{\cdot}\Phi_0{\cdot}\Phi_1 \cdots \Phi_i(2j^*)]_{n-1:1}.$$

PROOF: It is clear that $\Phi_i^{-1}{\cdot}\Phi_{i-1}^{-1} \cdots \Phi_0^{-1}{\cdot}\alpha^{-1}(2j)$ or $\Phi_i^{-1}{\cdot}\Phi_{i-1}^{-1} \cdots \Phi_0^{-1}{\cdot}\alpha^{-1}(2j+1)$ represent two network inputs of MIN_1 which $SW_1[i, j]$ connected to. However, according to the criterion that two paths connecting the same source and the same destination of both MIN_1 and MIN_2 pass the same switching element at each stage, these two network inputs should pass the same switching element at stage i of MIN_2. Therefore, $j^* = [\Phi_i^{-1}{\cdot}\Phi_{i-1}^{-1} \cdots \Phi_0^{-1}{\cdot}\alpha^{-1}{\cdot}\Psi_0{\cdot}\Psi_1 \cdots \Psi_i(2j)]_{n-1:1}$ is the relative address of $SW_1[i, j]$ at stage i of MIN_2. Similarly, $j = [\Psi_i^{-1}{\cdot}\Psi_{i-1}^{-1} \cdots \Psi_0^{-1}{\cdot}\alpha{\cdot}\Phi_0{\cdot}\Phi_1 \cdots \Phi_i(2j^*)]_{n-1:1}$ is the relative address of $SW_2[i, j^*]$ at stage i of the transformed MIN_1.

□

For example, to know the relative position of each switching element of the transformed MIN_1 in Fig. 3.4 (which is transformed from an Omega network) with respect to that of MIN_2 in Fig. 3.2, we only have to check their transition sequences.

For MIN_2 in Fig. 3.2, the transition sequence is:

$$\begin{aligned}
S &= (s_3, s_2, s_1, s_0) \\
S^0 &= (s_2, \bar{s}_1, \bar{s}_0, s_3) \\
D^0 &= (s_2, \bar{s}_1, \bar{s}_0, d_2) \\
S^1 &= (\bar{s}_1, s_2, d_2, s_0) \\
D^1 &= (\bar{s}_1, s_2, d_2, \bar{d}_3) \\
S^2 &= (d_3, d_2, \bar{s}_1, \bar{s}_2) \\
D^2 &= (d_3, d_2, \bar{s}_1, \bar{d}_0) \\
S^3 &= (d_2, d_0, d_3, \bar{s}_1) \\
D^3 &= (d_2, d_0, d_3, d_1) \\
S^4 &= (d_3, d_2, d_1, d_0) \\
&= D.
\end{aligned}$$

For the transformed MIN_1 in Fig. 3.4, the transition sequence is:

$$\begin{aligned}
S &= (s_3, s_2, s_1, s_0) \\
S^0 &= (s_0, \bar{s}_2, \bar{s}_1, s_3)
\end{aligned}$$

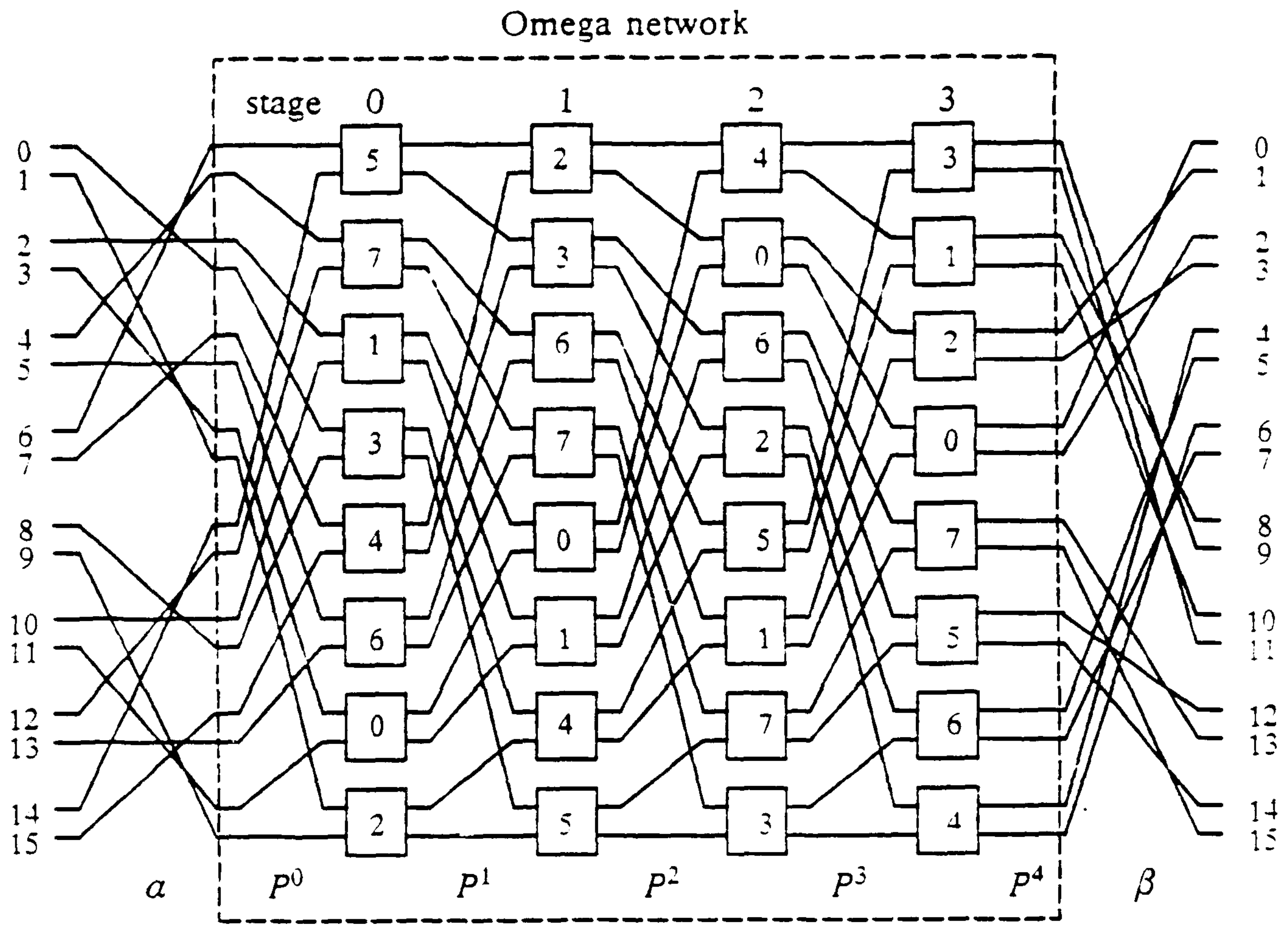


Fig. 3.6. The relative positions of switching elements of the transformed MIN_1 in Fig. 3.4 with respect to those of MIN_2 in Fig. 3.2.

$$D^0 = (s_0, \bar{s}_2, \bar{s}_1, d_2)$$

$$S^1 = (\bar{s}_2, \bar{s}_1, d_2, s_0)$$

$$D^1 = (\bar{s}_2, \bar{s}_1, d_2, \bar{d}_3)$$

$$S^2 = (s_1, d_2, \bar{d}_3, \bar{s}_2)$$

$$D^2 = (s_1, d_2, \bar{d}_3, \bar{d}_0)$$

$$S^3 = (d_2, \bar{d}_3, \bar{d}_0, \bar{s}_1)$$

$$D^3 = (d_2, \bar{d}_3, \bar{d}_0, d_1)$$

$$S^4 = (d_3, d_2, d_1, d_0)$$

$$= D .$$

Thus, at stage 0, the switching element $SW_1[0, j] = SW_1[0, (j_2,j_1,j_0)]$ in the transformed MIN_1 is the corresponding switching element $SW_2[0, j^*] = SW_2[0, (\bar{j}_1,j_0,\bar{j}_2)]$ in MIN_2, i.e., $SW_1[0,0] \rightarrow SW_2[0,5]$, $SW_1[0,1] \rightarrow SW_2[0,7]$, $SW_1[0,2] \rightarrow SW_2[0,1]$, $SW_1[0,3] \rightarrow SW_2[0,3]$, $SW_1[0,4] \rightarrow SW_2[0,4]$, $SW_1[0,5] \rightarrow SW_2[0,6]$, $SW_1[0,6] \rightarrow SW_2[0,0]$, and $SW_1[0,7] \rightarrow SW_2[0,2]$. Similarly, at other stages, we have $SW_1[1, (j_2,j_1,j_0)] \rightarrow SW_2[1, (j_1,\bar{j}_2,\bar{j}_0)]$, $SW_1[2, (j_2,j_1,j_0)] \rightarrow SW_2[2, (\bar{j}_0,j_1,\bar{j}_2)]$, and $SW_1[3, (j_2,j_1,j_0)] \rightarrow SW_2[3, (j_2,\bar{j}_0,\bar{j}_1)]$. The relative positions are shown in Fig. 3.6.

3.4. SUMMARY

In this chapter, the transformation rules for a MIN to simulate another is presented. The relative positions of each switching element before and after a network transformation are also described. Both distributing and global routing schemes are shown to be the same as the original MIN. By using the results presented in this chapter, the parallel algorithms developed for a MIN can be directly be reused on another MIN such that programming effort can be greatly reduced.

CHAPTER 4

PERMUTATION CAPABILITY OF MULTISTAGE INTERCONNECTION NETWORKS

4.1. INTRODUCTION

Various properties of the shuffle-exchange type multistage interconnection networks [WuFe81] have attracted considerable interest over the past decade. Particularly, a number of authors [Law75] [Ste83] [NaSa81] [Len78] [Sto71] have shown that these networks can perform a wide variety of useful permutations for parallel processing. A permutation is called *admissible* on a network iff it can be realized by one pass through the network without conflict at any switching element(s). One of the most important tasks in designing a parallel supercomputer is the selection of a suitable network in order to optimally support application needs. Before that, we need to be able to understand the permutation capability of each network. The set of admissible permutations of an Omega network has been characterized in [Law75] and [Par80], and later expressed more formally in [Pea77]. In their studies, the characterization of the admissible permutations is expressed by Boolean functions or bit relations of source tags and destination tags. However, from the viewpoint of applications, their results did not give any algorithm with low time complexity to determine the admissibility of a permutation. On the other hand, in the study of Lee [Lee85], the set of admissible permutations of the inverse Omega network has been characterized by using residue classes of destination tags. However, her analysis is rather tedious and indirect due to ignoring the characteristics of the structure of inverse Omega networks. Her result also suffers the problem of high complexity due to the use of modulo operations. Other than these studies, the characterization of admissible permutations of networks has seldom been mentioned.

While it has been proved that there exists a class of topologically equivalent networks with the same hardware complexity [Agr83], very little has been known about what kind of models can be used to characterize them. In this chapter, we introduce a general model. The characteristic of the permutation capability of a class of useful networks defined by this general model, which includes the six famous networks in [WuFe81] as special cases, is studied. Our analysis is based on the natural structure of a network which can be specified by two permutation functions. We start our discussion on Omega networks due to their regular structure, and then generalize the problem to the general model using bit-permute-complement connections. Our analysis is more direct, simple and general than all the previous works . We show that the set of admissible permutations of a network can be characterized by very simple bit relations depending on two permutation functions which specify this network. Our result shows that the time complexity of our proposed algorithm to determine the admissibility of a permutation on a network is $O(N)$, where N is the number of inputs/outputs of the network.

The remainder of this chapter is organized as follows. In Section 4.2, the basic definitions and notations are introduced. Particular attention is devoted to the routing behavior of Omega networks. In Section 4.3, by introducing a partitioning scheme, a sequence of substructures (subnetworks) are produced. These substructures are associated with some specific partitions on network inputs which can be used to characterize admissible permutations of an Omega network. The characteristic of admissible permutations of Omega networks is given in Section 4.4. In Section 4.5, a general model of a class of networks is defined. We show that our analytic methodology can be easily generalized to the general model. Finally, conclusions are given in Section 4.6.

4.2. PRELIMINARY

4.2.A. Omega Networks

In this chapter, without loss of generality, we start our discussion on the permutation capability of Omega networks [Law75] built with 2×2 switching elements. The general problem of various multistage interconnection networks is discussed in Section 4.5. An $N \times N$ Omega network consists of $n = \log_2 N$ stages of 2×2 switching elements for connecting N network inputs and N network outputs. (Note that, for simplicity, $\log_2 N$ is also denoted as $\log N$ in this chapter.) Each stage consists of $N/2$ switching elements and the interconnection pattern between stages is the perfect shuffle permutation. An Omega network for $N = 16$ is shown in Fig. 4.1. The following conventional notations are used throughout this chapter. The stages of the network are numbered from 0 through $n-1$ from left to right. The the input/output ports (including network inputs/outputs) of switching elements at each stage are numbered from 0 through $N-1$ and the switching elements, from 0 through $N/2 - 1$ from top to bottom. The binary representation of a number $l = (l_{n-1}, \ldots, l_1, l_0)$ (where bit l_{n-1} is the the most significant bit (MSB) and bit l_0, the least significant bit (LSB)) is used to represent the address of this number. A set of numbers with a similar address representation can be represented by a common address label. For example, $(l_{n-1}, \ldots, l_i, c, \ldots, c)$, where $c = 0$ or 1

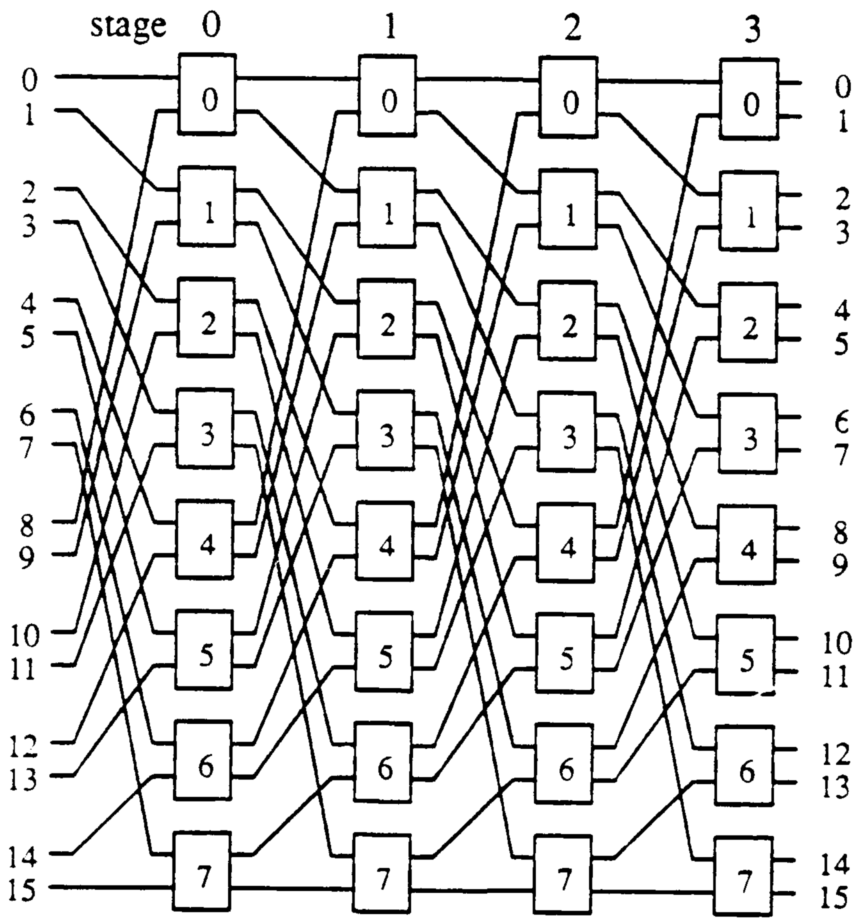


Fig. 4.1. A 16 × 16 Omega network.

(i.e., don't care) and $\#(c) = i$, (i.e., the total number of c's is equal to i) represents those 2^i numbers with the same first $n - i$ bits in their addresses. The notation $[l]_{a:b}$ is used to represent a segment of the address l from bit l_a to bit l_b, i.e., $(l_a, l_{a-1}, \ldots, l_b)$. If $a = b$, then $[l]_a$ denotes bit l_a in the binary representation of l. (Throughout this chapter, it is assumed that all the variables are integers.) A simple routing scheme on an Omega network can be described as follows. Let $D(i)$ be the destination tag of a data packet from network input $i = (i_{n-1}, \ldots, i_1, i_0)$, $0 \le i \le 2^n - 1$. That is, this data packet from network input i will be routed to the network output $D(i)$. Then, according to the routing scheme of Omega networks [Law75], bit $[D(i)]_{n-1-j}$ is used to determine the connection of the switching element at stage j, $0 \le j \le n - 1$, on the path connecting input i to output $D(i)$.

If $D(i) \ne D(k)$ for $i \ne k$ and all $0 \le i, k \le N - 1$, then, $D = (D(0), D(1), \ldots, D(N-1))$ *represents a permutation of* $(0, 1, \ldots, N-1)$. There are totally $N!$ different permutations of $(0, 1, \ldots, N-1)$ and we denote them as the set $\{\Xi_N\}$. Let Ω denote the set of all the admissible permutations of an Omega network. Since an Omega network contains $(N \log N)/2$ switching elements, each of which can be set in either one of the two states (i.e., either straight connection or crossing connection), different settings of these switching elements pass different permutations. It can be easily proved that $\#(\Omega) = 2^{\frac{N}{2}\log N} = N^{N/2}$.

It is convenient to describe some frequently used permutations. One of them is the family of *Bit-Permute-Complement* (BPC) type permutations, denoted by $\{\beta_n\}$.

DEFINITION 1: Let $l = (l_{n-1}, \ldots, l_1, l_0)$ be a number in $\{0, 1, \ldots, N-1\}$. A permutation $\beta \in \{\beta_n\}$ is specified by an n-tuple vector $\hat{\beta} = (\lambda_{n-1}\theta(n-1), \ldots, \lambda_1\theta(1), \lambda_0\theta(0))$, where $(\theta(n-1), \ldots, \theta(1), \theta(0))$ is a permutation of $(n-1, \ldots, 1, 0)$ and $\lambda_i \in \{-1, 1\}$, $0 \le i \le n - 1$, such that $\beta(l_{n-1}, \ldots, l_1, l_0) = (m_{n-1}, \ldots, m_1, m_0)$, where $m_i = l_{\theta(i)}$ if $\lambda_i = 1$, else $m_i = 1 - l_{\theta(i)} = \bar{l}_{\theta(i)}$ if $\lambda_i = -1$. □

In other words, $\beta(l)$ is obtained from l by first permuting the bits of the binary representation of l and then complementing a subset of bits according to the vector $\hat{\beta}$. Similarly, the inverse function β^{-1} and the absolute function $|\beta|$ of β are defined as follows.

DEFINITION 2: Let β^{-1} be the inverse function of $\beta \in \{\beta_n\}$. The function $\beta^{-1} \in \{\beta_n\}$ is specified by an n-tuple vector $\hat{\beta}^{-1} = (\lambda_{\theta^{-1}(n-1)}\theta^{-1}(n-1), \ldots, \lambda_{\theta^{-1}(1)}\theta^{-1}(1), \lambda_{\theta^{-1}(0)}\theta^{-1}(0))$ where θ^{-1} is the inverse function of θ. □

DEFINITION 3: Let $|\beta|$ be the absolute function of $\beta \in \{\beta_n\}$. The function $|\beta| \in \{\beta_n\}$ is specified by an n-tuple vector $|\hat{\beta}| = (|\lambda_{n-1}|\theta(n-1), \ldots, |\lambda_1|\theta(1), |\lambda_0|\theta(0))$. □

Let α and β be two permutation functions in $\{\beta_n\}$ and be specified by vectors $\hat{\alpha}$ and $\hat{\beta}$, respectively. The composition of α and β is denoted as $\alpha \cdot \beta$ such that $\alpha \cdot \beta(l) = \alpha(\beta(l))$ (i.e., the composed functions are performed from right to left) and is specified by a vector $\hat{\alpha} \cdot \hat{\beta}$. For example, consider $n = 4$ and let β be specified by $\hat{\beta} = (-2, -1, 0, 3)$ and $l = (1, 0, 0, 1)$. We have $m_3 = 1 - l_2$, $m_2 = 1 - l_1$, $m_1 = l_0$ and $m_0 = l_3$. Hence, $\beta(l) = (1, 1, 1, 1)$. Similarly, it is easy to obtain that $\hat{\beta}^{-1} = (0, -3, -2, 1)$, $\beta^{-1}(l) = (l_0, 1 - l_3, 1 - l_2, l_1) = (1, 0, 1, 0)$ and $|\hat{\beta}| = (2, 1, 0, 3)$, $|\beta|(l) = (0, 0, 1, 1)$. If α is specified by $\hat{\alpha} = (-1, 2, 3, -0)$, then $\alpha \cdot \beta(l) = \alpha(\beta(l)) = \alpha(1, 1, 1, 1) = (0, 1, 1, 0)$ and the composition $\alpha \cdot \beta$ is specified by the vector $\hat{\alpha} \cdot \hat{\beta} = (-1, 2, 3, -0) \cdot (-2, -1, 0, 3) = (-0, -1, -2, -3)$. Also note that the *bit-reversal* permutation ρ is one of the BPC type permutations such that $\rho(l) = (l_0, l_1, \ldots, l_{n-2}, l_{n-1})$, where $\hat{\rho} = (0, 1, \ldots, n-1)$. Another example is the perfect shuffle permutation ζ such that $\zeta(l) = (l_{n-2}, \ldots, l_0, l_{n-1})$, where $\hat{\zeta} = (n-2, \cdots, 0, n-1)$.

4.2.B. Routing Behavior of Omega Networks

For an n-stage Omega network, due to its regular structure, any path connecting network input $s = (s_{n-1}, ..., s_1, s_0)$ to network output $D(s) = (d_{n-1}, ..., d_1, d_0)$, can be expressed by the following transition sequence:

$$s = (s_{n-1}, s_{n-2}, ..., s_1, s_0)$$

$$s^0 = (s_{n-2}, s_{n-3}, ..., s_0, s_{n-1})$$

$$D^0(s) = (s_{n-2}, s_{n-3}, ..., s_0, d_{n-1})$$

$$s^1 = (s_{n-3}, s_{n-4}, ..., s_0, d_{n-1}, s_{n-2})$$

$$D^1(s) = (s_{n-3}, s_{n-4}, ..., s_0, d_{n-1}, d_{n-2})$$

......

$$s^i = (s_{n-2-i}, s_{n-3-i}, ..., s_0, d_{n-1}, \ldots d_{n-i}, s_{n-1-i})$$

$$D^i(s) = (s_{n-2-i}, s_{n-3-i}, ..., s_0, d_{n-1}, ..., d_{n-i}, d_{n-1-i})$$

......

$$s^{n-1} = (d_{n-1}, d_{n-2}, ..., d_1, s_0)$$

$$D^{n-1}(s) = (d_{n-1}, d_{n-2}, ..., d_1, d_0)$$

$$s^n = (d_{n-1}, d_{n-2}, ..., d_1, d_0)$$

$$= D(s).$$

In the transition sequence, each s^i, $0 \le i \le n-1$, represents the address of the input port of a switching element at stage i through which a path starting from s traverses stage i and each $D^i(s)$, $0 \le i \le n-1$, the output port of the same switching element through which the path traverses stage i. That is, a data transfer path of this switching element is connected from input port s^i to output port $D^i(s)$. Obviously, $[s^i]_{n-1:1} = [D^i(s)]_{n-1:1}$ is the address of this switching element through which a path traverses stage i. Similarly, the idea of the transition sequence can be used at each stage to express paths which connect a switching element to network inputs and outputs. Assume that a switching element E at stage i has the address $E = (e_{n-1}, \ldots, e_1)$ and the address label of input/output ports E is $e = (e_{n-1}, \ldots, e_1, c)$. We have the following two transition sequences to indicate which network input s's and output $D(e)$'s are connected through E.

Backward:

$$e = (e_{n-1}, e_{n-2}, \ldots, e_1, c)$$

$$e^i = (c, e_{n-1}, \ldots, e_2, e_1)$$

$$D^{i-1}(e) = (c, e_{n-1}, \ldots, e_2, c)$$

$$e^{i-1} = (c, c, e_{n-1}, \ldots, e_2)$$

......

$$D^0(e) = (c, \ldots, c, e_{n-1}, \ldots, e_{i+1}, c)$$

$$e^0 = (c, \ldots, c, e_{n-1}, \ldots, e_{i+2}, e_{i+1})$$

$$= s;$$

Forward:

$$e = (e_{n-1}, e_{n-2}, \ldots, e_1, c)$$

$$e^{i+1} = (e_{n-2}, e_{n-3}, \ldots, e_1, c, e_{n-1})$$

$$D^{i+1}(e) = (e_{n-2}, e_{n-3}, \ldots, e_1, c, c)$$

......

$$e^{n-1} = (e_i, \ldots, e_1, c, \ldots, c, e_{i+1})$$

$$D^{n-1}(e) = (e_i, \ldots, e_1, c, \ldots, c, c)$$

$$e^n = (e_i, \ldots, e_1, c, \ldots, c, c)$$

$$= D(e).$$

The switching element E at stage i can be viewed as the common root of two communication binary trees. One is the backward $(i+1)$-level tree with the address label of its leaves (i.e., switching elements at stage 0) equal to $(c, \ldots, c, e_{n-1}, \ldots, e_{i+1})$, $\#(c) = i$, and the address label of network inputs connected to leaves equal to $s = (c, \ldots, c, e_{n-1}, \ldots, e_{i+1})$, $\#(c) = i + 1$. The other one is the forward $(n-i)$-level tree with the address label of its leaves (i.e., switching elements at stage $n-1$) equal to $(e_i, \ldots, e_1, c, \ldots, c)$, $\#(c) = n - i - 1$, and the address label of network outputs connected to leaves equal to $D(e) = (e_i, \ldots, e_1, c, \ldots, c)$, $\#(c) = n - i$. Thus, totally 2^{i+1} network inputs and 2^{n-i} network outputs are connected through E. Particularly, when $i = 0$ $(n - 1)$, these above two binary trees are reduced to one in which the root E is rooted at stage 0 $(n-1)$ and the root E is connected to two network inputs $s = (c, e_{n-1}, \ldots, e_1)$ (two network outputs, $D(e) = (e_{n-1}, \ldots, e_1, c)$) and all the network outputs (inputs).

4.3. PERMUTABLE STRUCTURE

There are many different ways to partition an Omega network into disjoint subnetworks by forcing all the switching elements of one or more stages to straight connection (0-state) or crossing connection (1-state). Any switching element forced to a fixed state can be removed and replaced by two direct connecting links between its input and output ports. In this section, by introducing a proper partitioning scheme, a sequence of substructures of an Omega network, referred to as *permutable* substructures, are produced. Each substructure is a subnetwork which can be used to characterize admissible permutations of an Omega network. Our work is based on the following fact.

THEOREM 1: By forcing all the switching elements at stage i to 0-state or 1-state of an n-stage Omega network, two disjoint $(n-1)$-stage subnetworks are formed such that the $(n-i-1)$th bits of the input or output addresses in each subnetwork are the same.

PROOF: Proof can be given by referring to the transition sequence described in Section 4.2. By observing the numbers s^i and $D^i(s)$, the following fact can be obtained. Forcing all the switching elements at stage i to 0-state (1-state) is equivalent to forcing the LSB, s_{n-1-i}, of s^i to be replaced by s_{n-1-i} $(1 - s_{n-1-i})$ in the LSB position of $D^i(s)$.

Let us consider the case where switching elements at stage i are forced to 0-state. In each switching element at stage i, a data packet is forced to be routed from the s_{n-1-i} input port to the $d_{n-1-i} = s_{n-1-i}$ output port. That is to say, any input s can only communicate with an output $D(s)$ with the bit $d_{n-1-i} = s_{n-1-i}$. Obviously, two subnetworks are formed by partitioning the N network inputs and outputs into two groups such that in each subnetwork the addresses of the $N/2$ network inputs agree in their $(n-1-i)$th bits (i.e., s_{n-1-i}'s), the addresses of the $N/2$ network outputs agree in their $(n-1-i)$th bits (i.e., d_{n-1-i}'s) and s_{n-1-i} $= d_{n-1-i}$. To prove that these two subnetworks are disjoint, it is sufficient to prove that there are no common switching elements on these two subnetworks. Assume that Γ_0 represents the subnetwork with $s_{n-1-i} = d_{n-1-i} = 0$ in addresses of its network inputs/outputs and Γ_1, the subnetwork with $s_{n-1-i} = d_{n-1-i} = 1$ in addresses of its network inputs/outputs. Recall that either $[s^j]_{n-1:1}$ or $[D^j(s)]_{n-1:1}$ is the address of the the switching element through which the path connecting s to $D(s)$ traverses stage j, $0 \leq j \leq n-1$. By observing any transition sequences on Γ_0 and Γ_1, it is easy to see that at any stage $j \neq i$, any switching element $[s^j]_{n-1:1}$ (or $[D^j(s)]_{n-1:1}$) of Γ_0 is different from any switching element of Γ_1 in at least one bit position where either bit s_{n-1-i} or d_{n-1-i} appears. This means no common switching elements exist on Γ_0 and Γ_1. Thus, Γ_0 and Γ_1 are disjoint.

Similarly, for the 1-state case, two disjoint subnetworks can be formed by partitioning the N network inputs and outputs into two groups such that in each subnetwork the addresses of the $N/2$ network inputs agree in their $(n-1-i)$th bits (i.e., s_{n-1-i}'s), the addresses of the $N/2$ *network outputs agree in their* $(n-1-i)$th bits (i.e., d_{n-1-i}'s) and $s_{n-1-i} = 1 - d_{n-1-i}$. □

For example, by forcing all the switching elements at stage 1 of the Omega network shown in Fig. 4.1 to 0-state, two disjoint 3-stage subnetworks are formed. In one of them, the addresses of the eight network inputs agree in bit $s_2 = 0$ (i.e., they are {0, 1, 2, 3, 8, 9, 10, 11}) and the eight network outputs, in bit $d_2 = 0$ (i.e., they are {0, 1, 2, 3, 8, 9, 10, 11}). In the other one, the addresses of the eight network inputs agree in bit $s_2 = 1$ and the addresses of the eight network outputs agree in bit $d_2 = 1$. These two subnetworks are shown in Fig. 4.2.

Now, we employ a partitioning scheme on Omega networks which gives a better analytic way than that in [Lee85] in order to have a global view on the permutation behavior of Omega networks. The partitioning scheme which can produce a sequence of substructures on

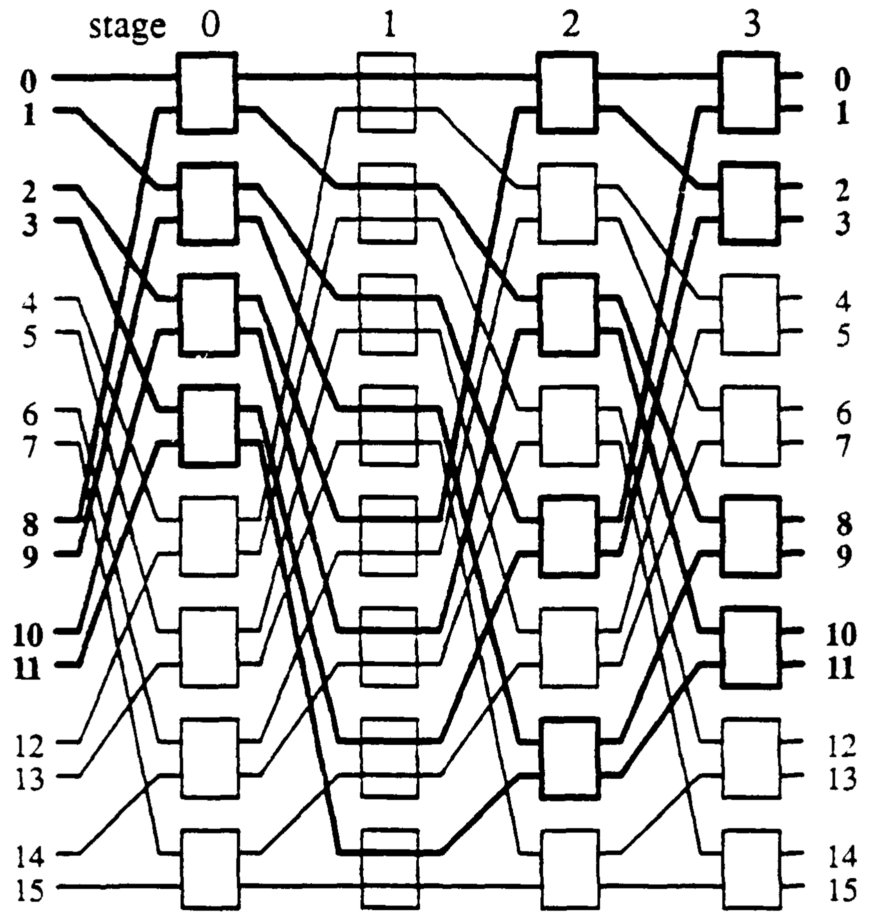


Fig. 4.2. Two 3-stage subnetworks are formed by forcing all the switching elements at stage 1 on a 16×16 Omega network.

an Omega network is described as follows. According to Theorem 1, if we remove stage $n-1$ of an Omega network, then two $(n-1)$-stage disjoint subnetworks will be produced. Here, by removing the last stage from a (sub)network we mean that both the last stage and the connection pattern before the stage are removed and thus the remaining output ports are left as network outputs of the two disjoint subnetworks. If we remove stage $n-2$ of any one of these $(n-1)$-stage subnetworks, then another two $(n-2)$-stage disjoint subnetworks will be produced. This process can be continued by removing stage $i-1$ of any i-stage subnetwork to produce another two $(i-1)$-stage disjoint subnetworks, for all $2 \leq i \leq n - 1$. When $i = 2$, after removing stage 1, subnetworks with single switching element will be produced. The above argument implies a recursively partitionable structure of Omega networks. We specify it by the following definition and theorems.

DEFINITION 4: Let $\Phi[n - 1, 0]$ be an Omega network and $\Phi[n - 2, t]$, $0 \leq t \leq 1$, be an $(n-1)$-stage subnetwork produced by removing stage $n-1$ of $\Phi[n - 1, 0]$. The $(i+1)$-stage subnetwork $\Phi[i, t]$, $0 \leq i \leq n - 2$ and $0 \leq t \leq 2^{n-i-1} - 1$, is obtained by removing stage $i+1$ of an $(i+2)$-stage subnetwork $\Phi[i + 1, t']$, $0 \leq t' \leq 2^{n-i-2} - 1$. Let $min(\Phi[i, t])$ be the smallest address of switching elements at stage i of subnetwork $\Phi[i, t]$. We assume that for any two subnetworks $\Phi[i, t^*]$ and $\Phi[i, t^{**}]$, $t^* > t^{**}$ iff $min(\Phi[i, t^*]) > min(\Phi[i, t^{**}])$. □

The following theorem shows which network inputs are connected to a subnetwork $\Phi[i, t]$.

THEOREM 2: Let $\Psi(i, t)$ be the set of network inputs connected to $\Phi[i, t]$, where $0 \leq i \leq n - 1$, $0 \leq t \leq 2^{n-i-1} - 1$ and $t = (t_{n-i-2}, \ldots, t_1, t_0)$ be the binary representation of t. Then, $\Psi(i, t) = \{(c, \ldots, c, t_{n-i-2}, \ldots, t_1, t_0) \mid \#(c) = i + 1\}$ and $\{(t_{n-i-2}, \ldots, t_1, t_0, c, \ldots, c) \mid$

$\#(c) = i\}$ is the set of switching elements at stage i of $\Phi[i, t]$. For $i = n - 1$, $\Phi[n - 1, 0]$ is an Omega network and $\Psi(n - 1, t) = \{0, 1, ..., N-1\}$.

PROOF: According to the partitioning scheme, the set of subnetworks $\{\Phi[i, t] \mid 0 \le t \le 2^{n-i-1} - 1\}$ is produced by removing all the stages from stage $n-1$ to stage $i+1$. And, according to Theorem 1, the last $n - i - 1$ bits (starting from LSB to the $(n-i-2)$th bit) of all the network inputs in set $\Psi(i, t)$ corresponding to the subnetwork $\Phi[i, t]$ are the same. Thus, $\Psi(i, t) = \{(c, ..., c, t'_{n-i-2}, ..., t'_1, t'_0) \mid \#(c) = i + 1\}$. To prove $(t'_{n-i-2}, ..., t'_1, t'_0) = (t_{n-i-2}, ..., t_1, t_0)$, let us check the following fact. Let $min(\Psi(i, t))$ be the smallest address in set $\Psi(i, t)$. Due to the inverse perfect shuffle connections, when one backtracks from stage $n-1$ of an Omega network, it is very easy to observe that $\{(t_{n-i-2}, ..., t_1, t_0, c, ..., c) \mid \#(c) = i\}$ is the set of switching elements at stage i of $\Phi[i, t]$. By backtracking $i + 1$ inverse perfect shuffle connections, we can also see that the set of switching elements $\{(t_{n-i-2}, ..., t_1, t_0, c, ..., c) \mid \#(c) = i\}$ is connected to the set of inputs $\{(c, ..., c, t_{n-i-2}, ..., t_1, t_0) \mid \#(c) = i + 1\}$. Thus, for any $t^* > t^{**}$, we always have $min(\Phi[i, t^*]) = (t^*_{n-i-2}, ..., t^*_1, t^*_0, 0, ..., 0) > min(\Phi[i, t^{**}]) = (t^{**}_{n-i-2}, ..., t^{**}_1, t^{**}_0, 0, ..., 0)$. That is, $(t^*_{n-i-2}, ..., t^*_1, t^*_0) > (t^{**}_{n-i-2}, ..., t^{**}_1, t^{**}_0)$ which in turn implies $min(\Psi(i, t^*)) > min(\Psi(i, t^{**}))$. □

For example, for $N = 16$, the three sets of subnetworks $\{\Phi[i, t] \mid 0 \le t \le 2^{3-i} - 1\}$, $0 \le i \le 2$, of an Omega network are shown in Fig. 4.3(a)(b)(c). In Fig. 4.3(a), the set of two subnetworks $\{\Phi[2, t] \mid 0 \le t \le 1\}$ is obtained by removing stage 3 from the Omega network. The set of network inputs $\Psi(2, 0)$ corresponding to the subnetwork $\Phi[2, 0]$ is $\{(c, c, c, 0)\} = \{0, 2, 4, 6, 8, 10, 12, 14\}$ and the set of network inputs $\Psi(2, 1)$ associated with the subnetwork $\Phi[2, 1]$ is $\{(c, c, c, 1)\} = \{1, 3, 5, 7, 9, 11, 13, 15\}$. Similarly, in Fig. 4.3(b) and 4.3(c), the sets of network inputs $\Psi(1, t)$ and $\Psi(0, t)$ corresponding to the subnetworks $\Phi[1,$

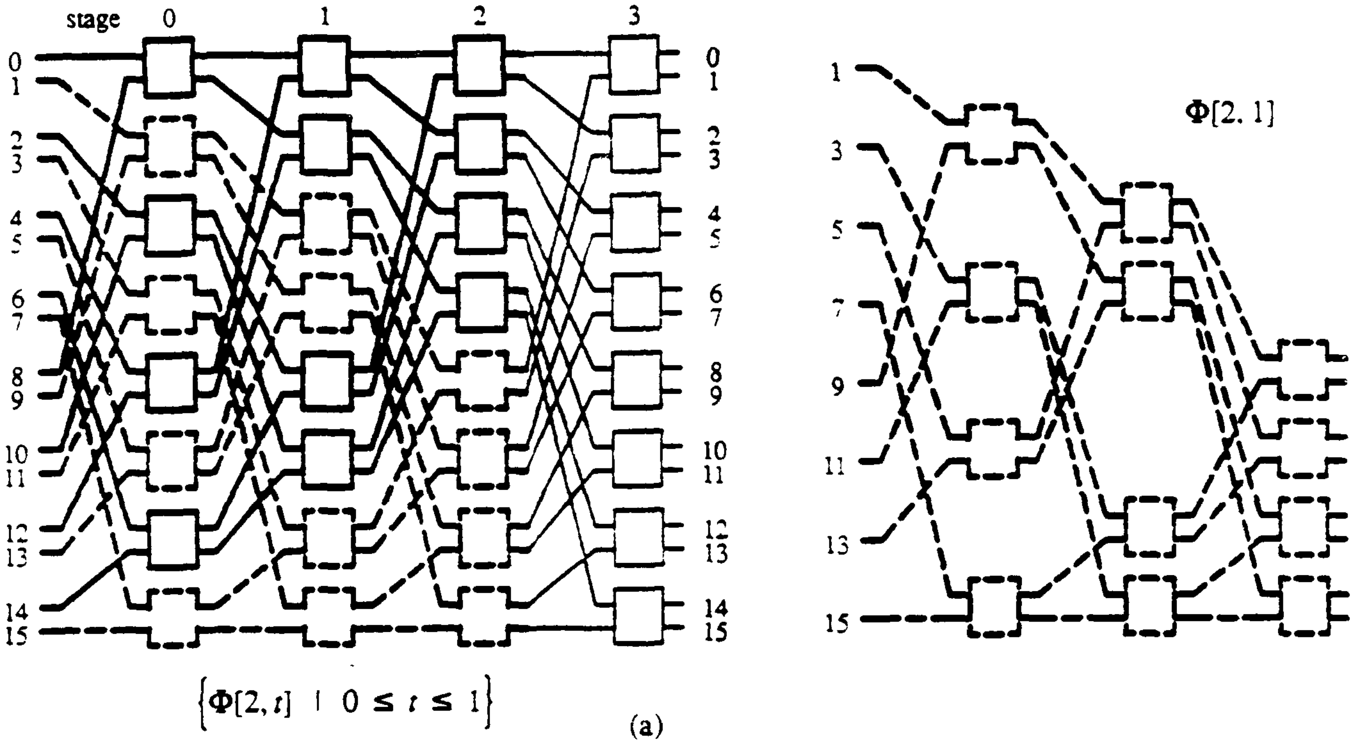
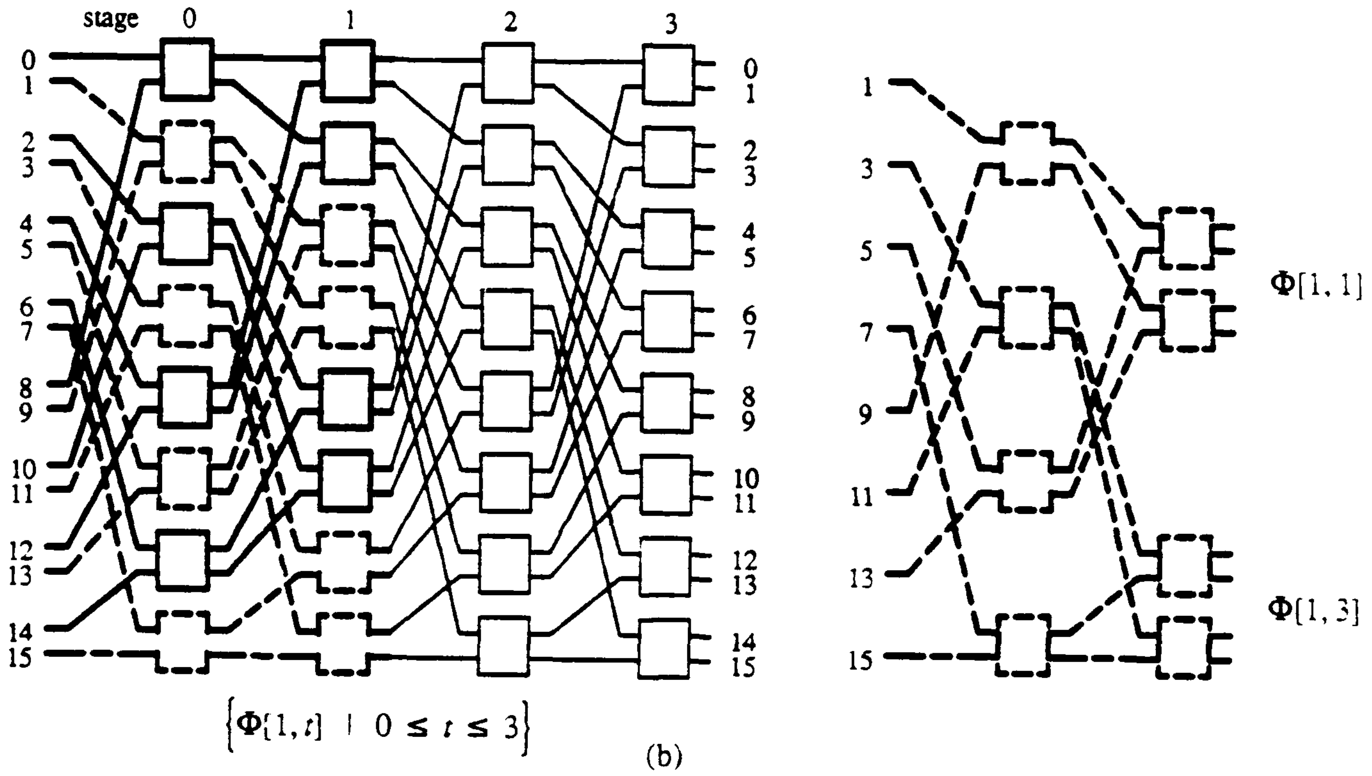
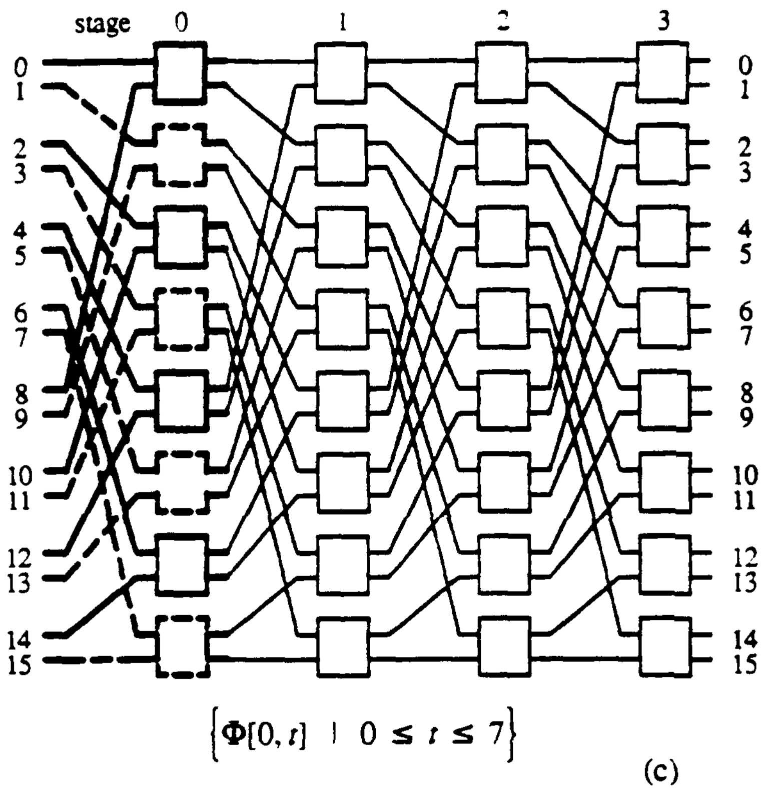


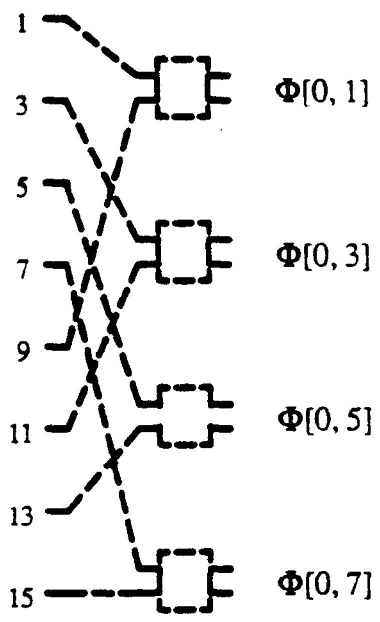
Fig. 4.3. For $N = 16$, the three sets of subnetworks $\{\Phi(i,t) \mid 0 \le t \le 2^{3-i}-1\}$, $0 \le i \le 2$, of an Omega network.



stage
0
1
2
3
0
1
2
3
4
5
6
7
8
9
10
11
12
13
14
15
$\{\Phi[1,t] \mid 0 \le t \le 3\}$
(b)
1
3
5
7
9
11
13
15
$\Phi[1,1]$
$\Phi[1,3]$

Fig. 4.3. Cont'd.





(c)

Fig. 4.3. Cont'd.

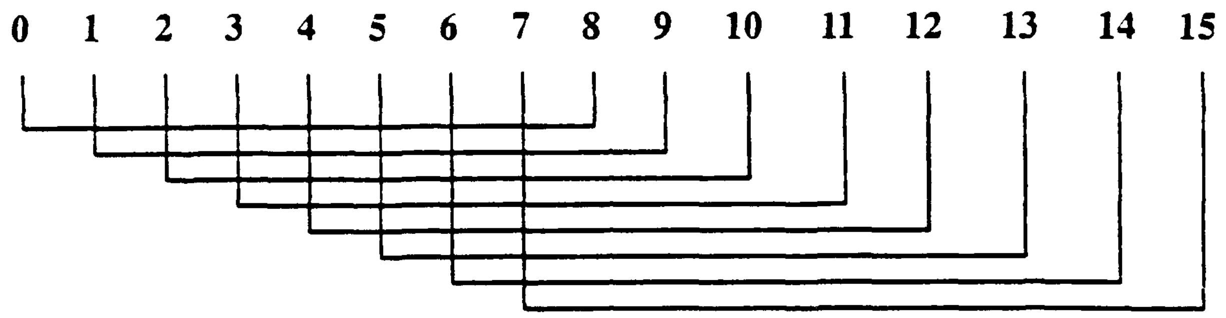
$t]$ and $\Phi[0, t]$ is $\{(c, c, t_1, t_0)\}$ and $\{(c, c, c, t_0)\}$, respectively.

Theorem 2 outlines the structure of each subnetwork in terms of the set of network inputs connected to it and the set of switching elements at the last stage of it. Obviously, because of the recursively partitionable structure of an Omega network, each subnetwork can also be recursively partitioned like an Omega network. This is based on the fact that the structure of any subnetwork is identical to that of an Omega network with reduced size. The following two theorems describe the substructure of a subnetwork after being recursively partitioned. Since their proof are similar to that of Theorem 2, we omit them here.

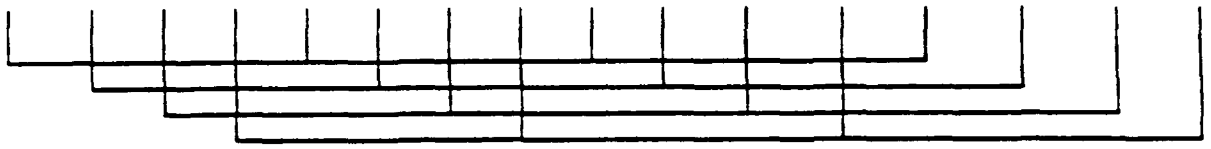
THEOREM 3: Two i-stage subnetworks $\{\Phi[i - 1, 2^{n-i-1} \cdot s + t] \mid 0 \le s \le 1\}$ are obtained by removing stage i of the $(i+1)$-stage subnetwork $\Phi[i, t]$, $1 \le i \le n - 1$ and $0 \le t \le 2^{n-i-1} - 1$. Each $\Phi[i - 1, 2^{n-i-1} \cdot s + t]$ is connected to the set of network inputs $\Psi(i - 1, 2^{n-i-1} \cdot s + t) = \{(c, ..., c, s, t_{n-i-2}, ..., t_1, t_0) \mid \#(c) = i\}$. □

For example, if we remove stage 2 of the 3-stage subnetwork $\Phi[2, 1]$ (i.e., $i = 2$ and $t = 1$), then we will obtain two 2-stage subnetworks $\{\Phi[1, 2 \cdot s + 1] \mid 0 \le s \le 1\}$. Each $\Phi[1, 2 \cdot s + 1]$ is connected to the set of network inputs $\Psi(1, 2 \cdot s + 1) = \{(c, ..., c, s, 1) \mid \#(c) = 2\}$. That is, $\Psi(1, 1) = \{1, 5, 9, 13\}$ and $\Psi(1, 3) = \{3, 7, 11, 15\}$.

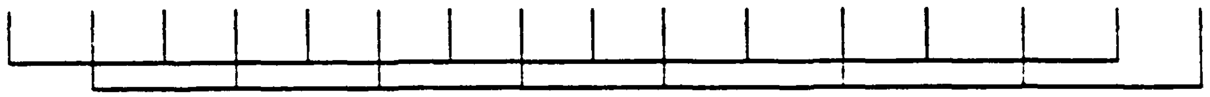
THEOREM 4: The set of $(j+1)$-stage subnetworks $\{\Phi[j, 2^{n-i-1} \cdot m + t] \mid 0 \le m \le 2^{i-j} - 1\}$, $0 \le j < i$, is obtained by removing stages from stage i to stage $j+1$ of the $(i+1)$-stage subnetwork $\Phi[i, t]$, $1 \le i \le n - 1$ and $0 \le t \le 2^{n-i-1} - 1$. Let $(m_{i-j-1}, ..., m_1, m_0)$ be the binary representation of m. Each $\Phi[j, 2^{n-i-1} \cdot m + t]$ is connected to the set of network inputs $\Psi(j, 2^{n-i-1} \cdot m + t) = \{(c, ..., c, m_{i-j-1}, ..., m_1, m_0, t_{n-i-2}, ..., t_1, t_0) \mid \#(c) = j + 1\}$□



$\left\{\Psi(0,t) \mid 0 \le t \le 7\right\} = \{\{0,8\}, \{1,9\}, \{2,10\}, \{3,11\}, \{4,12\}, \{5,13\}, \{6,14\}, \{7,15\}\}$



$\left\{\Psi(1,t) \mid 0 \le t \le 3\right\} = \{\{0,4,8,12\}, \{1,5,9,13\}, \{2,6,10,14\}, \{3,7,11,15\}\}$



$\left\{\Psi(2,t) \mid 0 \le t \le 1\right\} = \{\{0,2,4,6,8,10,12,14\}, \{1,3,5,7,9,11,13,15\}\}$

Fig. 4.4 For $N = 16$, the three different partitions $\left\{\Psi(i,t) \mid 0 \le t \le 2^{3-i} - 1\right\}$, $0 \le i \le 2$.

For example, for $i = 2$, $j = 0$ and $t = 1$, if we remove stages from stage 2 to stage 1 of the 3-stage subnetwork $\Phi[2, 1]$, then we will obtain the set of 1-stage subnetworks $\{\Phi[0, 2 \cdot m + 1] \mid 0 \leq m \leq 3\}$. Each $\Phi[0, 2 \cdot m + 1]$ is connected to the set of network inputs $\{(c, ..., c, m_1, m_0, 1) \mid \#(c) = 1\}$. That is, $\Psi(0, 1) = \{1, 9\}$, $\Psi(0, 3) = \{3, 11\}$, $\Psi(0, 5) = \{5, 13\}$, and $\Psi(0, 7) = \{7, 15\}$.

Note that, for each $0 \leq i \leq n - 1$, the set of subnetworks $\{\Phi[i, t] \mid 0 \leq t \leq 2^{n-i-1} - 1\}$ corresponds to the set of inputs $\{\Psi(i, t) \mid 0 \leq t \leq 2^{n-i-1} - 1\}$ which is a partition on all the network inputs $\{0, 1, ..., N-1\}$. For each i, the partition $\{\Psi(i, t)\}$ contains 2^{n-i-1} groups of network inputs. Each group of network inputs has 2^{i+1} elements. In the following sections, we will show that these partitions play a major role when we characterize Ω. For example, the three corresponding partitions $\{\Psi(i, t) \mid 0 \leq t \leq 2^{3-i} - 1\}$ associated with the three sets of subnetworks $\{\Phi[i, t] \mid 0 \leq t \leq 2^{3-i} - 1\}$, $0 \leq i \leq 2$, are shown in Fig. 4.4. When $i = 3$, $\{\Phi[i, 0]\}$ and $\{\Psi(i, 0)\}$ become trivial cases, i.e., an Omega network and all the network inputs $\{0, 1, ..., N-1\}$, respectively. For $i = 2$, 1, and 0, we have

$\{\Psi(2, t) \mid 0 \leq t \leq 1\} = \{\{0, 2, 4, 6, 8, 10, 12, 14\}, \{1, 3, 5, 7, 9, 11, 13, 15\}\}$,

$\{\Psi(1, t) \mid 0 \leq t \leq 3\} = \{\{0, 4, 8, 12\}, \{1, 5, 9, 13\}, \{2, 6, 10, 14\}, \{3, 7, 11, 15\}\}$,

$\{\Psi(0, t) \mid 0 \leq t \leq 7\} = \{\{0, 8\}, \{1, 9\}, \{2, 10\}, \{3, 11\}, \{4, 12\}, \{5, 13\}, \{6, 14\}, \{7, 15\}\}$.

4.4. PERMUTATION CAPABILITY

The set of admissible permutations Ω has been characterized by a number of authors [Law75],[Par80],[Pea77]. The following theorem summarizes their work in network admissibility.

Let $\pi \equiv D \in \{\Xi_N\}$ be a permutation of $(0, 1, ..., N-1)$ which is associated with routing tags $D = (D(0), D(1), ..., D(N-1))$. Each routing tag $D(i)$ is used by the network input $i = (i_{n-1}, ..., i_1, i_0)$ of an n-stage Omega network to route data packets to the network output $D(i)$. According to the routing scheme of an Omega network, bit $[\rho(D(i))]_j$ is used to determine the connection of the switching element at stage j, $0 \le j \le n-1$. (Note that, as defined earlier, ρ is a bit-reversal permutation.)

THEOREM 5: A permutation $\pi \in \Omega$ iff for each i and j, $0 \le i, j \le N-1$ and $i \ne j$, either one of the following two conditions is true:

(1) $([D(i)]_{n-1:b}, [i]_{b-1:0}) \ne ([D(j)]_{n-1:b}, [j]_{b-1:0})$, for all $1 \le b \le n-1$ (see [Law75]).

(2) There exists a Boolean function $f_b([D(i)]_{n-1:b+1}, [i]_{b-1:0})$ such that $[D(i)]_b = [i]_b$ @ $f_b([D(i)]_{n-1:b+1}, [i]_{b-1:0})$, for all $0 \le b \le n-2$, where @ is the exclusive-or operation (see [Par80][Pea77]). □

Theorem 5 characterizes Ω by using the bit relation of source tags and destination tags. However, according to condition (1) of Theorem 5, in order to know whether or not an arbitrary permutation belongs to Ω, computation must be performed for all the possible combinations of i, j, and b. An algorithm to determine the admissibility of a permutation using condition (1) of Theorem 5 is described as follows:

```
function ADMISSIBILITY-1 ($s \in \{0, ..., N-1\}$, $D$: permutation)
    for $b = n-1$ downto 1 do
        for each $Q = \{s \mid$ the sources share the same lower $b$ address bits$\}$ do
            if $DIFFERENCE(\{[D(s)]_{n-1:b} \mid s \in Q\}) = false$ then return $false$
    return $true$
```

The function *DIFFERENCE* is an algorithm to determine whether or not a finite number of integers are different. It can be shown that for problem size N the fastest algorithm employed for *DIFFERENCE* takes a time in $O(N\log N)$ (i.e., this is the same lowest bound as that of sorting problem). By using function *DIFFERENCE*, the algorithm *ADMISSIBILITY*-1 compares the difference of the higher address bits of those destinations (in set $\{[D(s)]_{n-1:b} \mid s \in Q\}$) whose corresponding sources (in set Q) share the same lower address bits. Since for each b, $1 \le b \le n-1$, there are 2^b different Q sets, the algorithm *ADMISSIBILITY*-1 uses function *DIFFERENCE* 2^{n-b} times.

Condition (2) of Theorem 5 is a reformulation of Condition (1) in order to show that a special set of permutations is admissible on the indirect binary cube network. For this special set of permutations, the Boolean function f_b is easy to identify. However, in general, it is extremely difficult (if not impossible) to determine whether or not there exists such an f_b which satisfies condition (2) for an arbitrary permutation. And the time complexity for using condition (2) to determine the admissibility will be much higher than that using condition (1). In this chapter, we will show that the set Ω can be characterized in a much easier way than that given by Theorem 5 such that a simple and low complexity algorithm can be developed to distinguish permutations in Ω from the others. This is our main work in this section.

A key idea used throughout this section is the residue system in number theorem [Lee85].

DEFINITION 5: A *complete residue system modulo m* (CRS(m)) is a set of m integers which contains exactly one element of each residue class mod m. $\square$

In other words, if every element of a CRS(m) is divided by m, each of the possible remainder value from 0 through $m - 1$ can be obtained. We have the following natural observation on the number system composed of non-negative integers. Any consecutive 2^k numbers form a CRS(2^k) which yields each remainder from 0 through $2^k - 1$ when divided by 2^k. If the same numbers are divided by 2^{k-1} instead, there will be a pair for each remainder from 0 through $2^{k-1} - 1$. Thus, a CRS(2^k) contains two representatives of each residue class mod 2^{k-1}, i.e., a CRS(2^k) can be partitioned into two CRS(2^{k-1})'s. Since there are two ways to choose each representative of a residue class mod 2^{k-1}, as many as 4^{k-1} different ways of partitioning can be made on a CRS(2^k). For example,

$$\mathrm{CRS}(8):\{7, 6, 0, 1, 3, 4, 5, 2\}$$

$$\equiv \mathrm{CRS}(4):\{7, 1, 4, 2\} \cup \mathrm{CRS}(4):\{6, 0, 3, 5\}$$

$$\equiv \mathrm{CRS}(4):\{3, 1, 0, 2\} \cup \mathrm{CRS}(4):\{7, 4, 6, 5\}$$

......

As pointed out in Section 4.3, for each $0 \le i \le n - 1$, the set $\{\Psi(i, t) \mid 0 \le t \le 2^{n-i-1} - 1\}$ is a partition on all the network inputs $\{0, 1, ..., N-1\}$ and therefore it corresponds to a partition on $\{D(j) \mid 0 \le j \le N - 1\}$. We will show that the CRS property of all these n partitions on $\{D(j) \mid 0 \le j \le N\text{-}1\}$ ensures that there will be no conflict in any switching element(s) when the permutation $\pi \equiv D$ is realized on an Omega network.

THEOREM 6: A permutation π is admissible on the subnetwork $\Phi[i, t]$, (i.e., it can be realized without conflicts) where $0 \le i \le n - 2$ and $0 \le t \le 2^{n-i-1} - 1$, iff $\{\rho(D(j)) \mid j \in \Psi(k, 2^{n-i-1} \cdot m + t)\}$ is a CRS(2^{k+1}), for all $0 \le k \le i$ and $0 \le m \le 2^{i-k} - 1$.

PROOF: According to the routing scheme of an Omega network, bit $[\rho(D(i))]_j$ is used to determine the connection of the switching element at stage j, $0 \le j \le n-1$. For any k and m, we can imagine the subnetwork $\Phi[k, 2^{n-i-1} \cdot m + t]$ of $\Phi[i, t]$ as an independent $(k+1)$-stage network in which the routing tag used on input $j \in \Psi(k, 2^{n-i-1} \cdot m + t)$ is $[(D(j)]_{n-1:n-k-1}$ or $[\rho(D(j))]_{k:0}$. If $\{[(D(j)]_{n-1:n-k-1} \mid j \in \Psi(k, 2^{n-i-1} \cdot m + t)\}$ or $\{[\rho(D(j))]_{k:0} \mid j \in \Psi(k, 2^{n-i-1} \cdot m + t)\}$ is not a CRS(2^{k+1}), then there exist at least two network inputs of the subnetwork $\Phi[k, 2^{n-i-1} \cdot m + t]$ such that data packets from which are sent to the same output. That is, for at least two $x, y \in \Psi(k, 2^{n-i-1} \cdot m + t)$ and $x \ne y$, $[\rho(D(x))]_{k:0} = [\rho(D(y))]_{k:0}$. Thus, it results in conflict in at least one switching element on $\Phi[k, 2^{n-i-1} \cdot m + t]$. This can be easily shown by an inductive method starting from $k = 0$ in which case $\Phi[k, 2^{n-i-1} \cdot m + t]$ is a single switching element. Therefore, for all $0 \le k \le i$ and $0 \le m \le 2^{i-k} - 1$, $\{\rho(D(j)) \mid j \in \Psi(k, 2^{n-i-1} \cdot m + t)\}$ must be CRS(2^{k+1})'s, iff π can pass the subnetwork $\Phi[i, t]$ without conflicts on switching elements. □

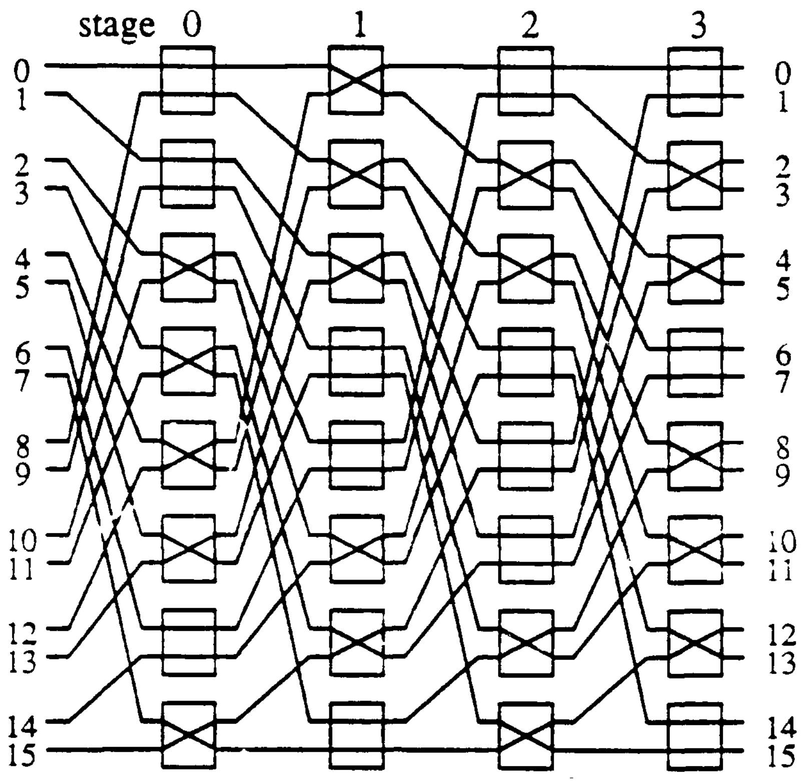
THEOREM 7: A permutation $\pi \in \Omega$ iff $\{\rho(D(j)) \mid j \in \Psi(i, t)\}$ is a CRS(2^{i+1}), for all $0 \le i \le n-1$ and $0 \le t \le 2^{n-i-1} - 1$.

PROOF: This proof is simply an extension from that of Theorem 6. When $i = n-1$, $\{\rho(D(j)) \mid j \in \Psi(n-1, 0)\} = \{0, 1, \ldots, N-1\}$ is a CRS(N) which is a trivial case and is always true. Thus, the permutation $\pi \in \Omega$ iff π is admissible on both subnetworks $\Phi[n-2, 0]$ and $\Phi[n-2, 1]$. □

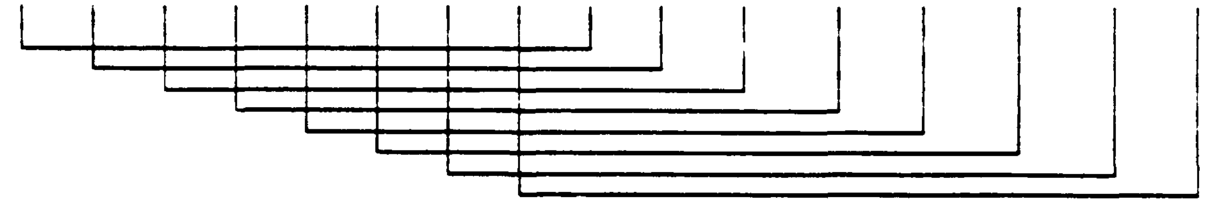
According to Theorem 7, a method is given to determine whether or not a given permutation is an admissible one of an Omega network. The work is composed of totally $\sum_{i=0}^{n-2} 2^{n-i-1}$ $= N/2 + N/4 + \cdots + 2 = N - 2$ subtasks and each subtask needs to determine the CRS property of a set of integer numbers (i.e., the set $\{\rho(D(j)) \mid j \in \Psi(i, t)\}$). Two example permutations are shown in Fig. 4.5 for a 16×16 Omega network. The first one is an admissible permutation (6, 4, 14, 8, 11, 15, 5, 12, 13, 10, 3, 7, 0, 1, 9, 2). The second one is a perfect shuffle permutation (0, 2, 4, 6, 8, 10, 12, 14, 1, 3, 5, 7, 9, 11, 13, 15) which is not admissible. For the first permutation, it has been shown that each $\{\rho(D(j)) \mid j \in \Psi(i, t)\}$, where $0 \le i \le 2$ and $0 \le t \le 2^{3-i} - 1$, is a CRS(2^{i+1}). To see why the second permutation is not admissible, let us check the subnetwork $\Phi[1, 0]$. The routing tags used for this subnetwork are $\{[\rho((D(j))]_{1:0} \mid j \in \Psi(1, 0)\} = \{[\rho((D(0))]_{1:0}, [\rho((D(4))]_{1:0}, [\rho((D(8))]_{1:0}, [\rho((D(12))]_{1:0}\}$ $= \{[0]_{1:0}, [1]_{1:0}, [8]_{1:0}, [9]_{1:0}\} = \{00, 01, 00, 01\} \neq$ CRS(4). This means that data packets from network inputs 0 and 8 are sent to the same network output 0 of $\Phi[1, 0]$. Similarly, data packets from network inputs 4 and 12 are sent to the same network output 1 of $\Phi[1, 0]$. They cause conflicts on switching elements of both stage 0 and stage 1 of $\Phi[1, 0]$.

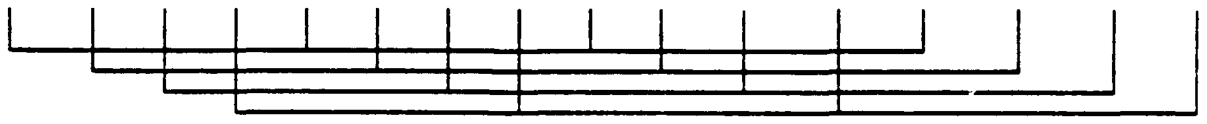
From Theorem 7, the set Ω is characterized by using the residue classes of destination tags rather than the bit relations of source tags and destination tags. However, as the result is compared to that of Theorem 5, we do not gain much in saving computational efforts since the modulo operations and the work to determine the CRS properties will consume a lot of time. The same problem was suffered in the work of Lee [Lee85]. Nevertheless, the result of Theorem 7 is still useful. We show next that a more effective way to characterize Ω than that of Theorem 5 can be derived from Theorem 7. Before we discuss that, let us see what the characteristics of Omega and inverse Omega networks are.

In the first transition sequence in Section 4.2, source bit s_{n-1-i} is moved to the LSB position of s^i at stage i and is replaced by the destination bit d_{n-1-i} of $D(s)$ in the LSB position of $D^i(s)$. That is, by the physical meaning, a data transfer path of the switching



s :	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
$D(s)$:	6	4	14	8	11	15	5	12	13	10	3	7	0	1	9	2
$\varrho(D(s))$:	6	2	7	1	13	15	10	3	11	5	12	14	0	8	9	4


CRS(2)'s


CRS(4)'s

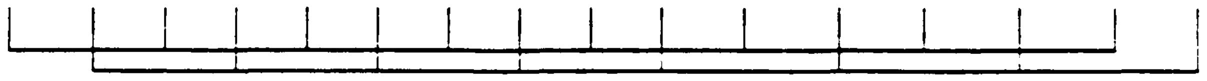

CRS(8)'s

Fig. 4.5. Two example permutations on a 16×16 Omega network.

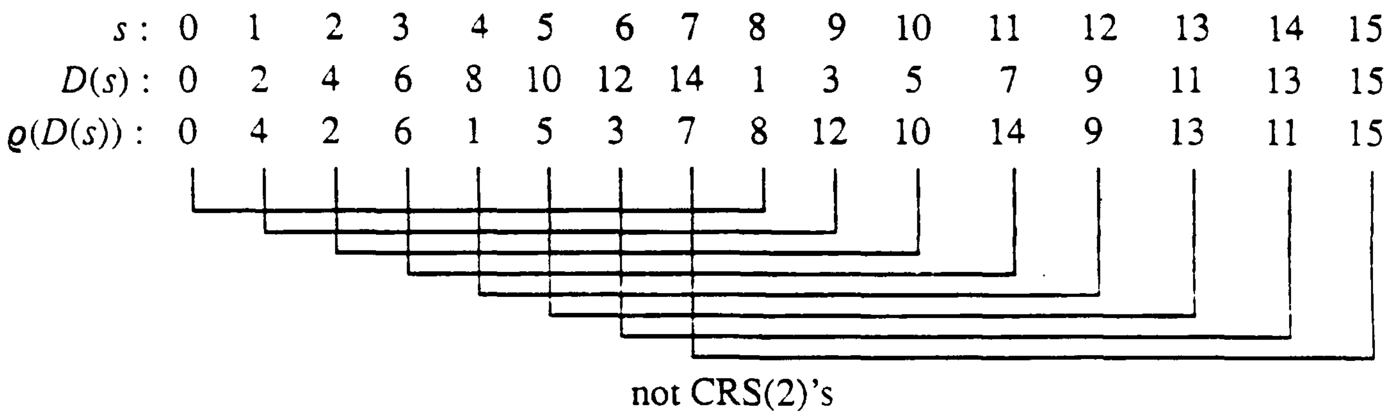

s : 0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15
$D(s)$: 0 2 4 6 8 10 12 14 1 3 5 7 9 11 13 15
$\varrho(D(s))$: 0 4 2 6 1 5 3 7 8 12 10 14 9 13 11 15
not CRS(2)'s

Fig. 4.5. Cont'd.

element $[s^i]_{n-1:1}$ (or $[D^i(s)]_{n-1:1}$) is connected from input port s_{n-1-i} to output port d_{n-1-i}. Therefore, the order for the source bits to be removed is $n-1$, $n-2$, $n-3$, ..., 2, 1, 0 and the order for the destination bits to be introduced is also $n-1$, $n-2$, $n-3$, ..., 2, 1, 0. We can use two vectors $\hat{O}$ = (0, 1, 2 , ..., $n-2$, $n-1$) and $\hat{I}^{-1}$ = (0, 1, 2 , ..., $n-2$, $n-1$) which correspond to two permutation functions O and I^{-1} in $\{\beta_n\}$, to represent these two sequences, respectively. They are referred to as *characteristic function* of an Omega network. Thus, by denoting

$$\hat{O} = (\ \hat{o}(n-1), \hat{o}(n-2)\ , \ldots\ , \hat{o}(2), \hat{o}(1), \hat{o}(0)\ )$$

and $$\hat{I}^{-1} = (\ \hat{i}^{-1}(n-1), \hat{i}^{-1}(n-2)\ , \ldots\ , \hat{i}^{-1}(2), \hat{i}^{-1}(1), \hat{i}^{-1}(0)\ )$$

we mean that at stage j source bit $s_{\hat{o}(j)}$ will be removed and replaced by destination bit $d_{\hat{i}^{-1}(j)}$. Moreover, the meaning of the permutation function I is as follows: if bit $s_{\hat{o}(j)}$ is replaced by bit d_j at stage j, then the order of bits of $I(D(s))$ represents the disturbed order of bits of $D(s)$. For an Omega network, we have $O = \rho$ and $I = I^{-1} = \rho$. From Theorems 1 and 2, it is obvious that the function $O = \rho$ can uniquely determine those n partitions $\{\Psi(i, t) \mid 0 \le t \le 2^{n-i-1} - 1\}$, $0 \le i \le n-1$. From Theorem 7, this in turn means that all the admissible permutations Ω can be uniquely characterized by functions O and I.

Let the characteristic functions of an inverse Omega network be denoted by O_R and I_R. Note that the $(n-i)$th stage of an Omega network becomes the ith stage of its inverse network. It is clear that any permutation $\pi \in \Omega$ iff π^{-1} is an admissible permutation of the inverse Omega network. We may denote the set of all the admissible permutations of an inverse Omega network by Ω^{-1}. Thus, for any transition sequence of an Omega network, we have a new explanation for its inverse Omega network: source bit d_{n-1-i} of $D(s)$ is moved to the LSB position of $D^i(s)$ at stage $n-i$ and is replaced by the destination bit s_{n-1-i} of s in the LSB position of s^i. That is, at stage j source bit $s_{\hat{o}(n-j)}$ will be removed and replaced by destination bit $d_{\hat{i}^{-1}(n-j)}$. Therefore, we have O_R and I_R^{-1} for an inverse Omega network as follows:

$$O_R = \rho \cdot I^{-1}, \quad \hat{O}_R = (n-1, n-2, ..., 1, 0)$$

$$I_R^{-1} = \rho \cdot O, \quad \hat{I}_R^{-1} = (n-1, n-2, ..., 1, 0).$$

Similarly, the function $O_R = \rho \cdot \rho$ which is an identity permutation uniquely determines those n partitions $\{\Psi_R(i, t) \mid 0 \leq t \leq 2^{n-i-1} - 1\}$, $0 \leq i \leq n - 1$. Thus, Ω^{-1} is uniquely characterized by functions O_R and I_R. Note that it is easy to prove that no two subnetworks $\Phi[a, b]$ and $\Phi_R[c, d]$ of an Omega network and its inverse Omega network respectively have the same set of switching elements at any stage.

The following theorem derived from Theorem 7 gives a simple closed form of Ω in terms of bit relations of destination tags with respect to permutable substructures of an Omega network and its inverse network.

THEOREM 8: A permutation $\pi \in \Omega$ iff $\sum_{j \in \Psi(i,t)} [\rho(D(j))]_i = \sum_{j \in \Psi_R(i,t)} [D^{-1}(j)]_i = 2^i$, for all $0 \leq i \leq n - 1$ and $0 \leq t \leq 2^{n-i-1} - 1$. That is, for both of all $j \in \Psi(i, t)$ and $j \in \Psi_R(i, t)$, the sum of ith bits of $\rho(D(j))$'s and $D^{-1}(j)$'s are both equal to 2^i.

PROOF: The proof is based on the following fact. Let $\{R_j \mid 0 \leq j \leq 2^{i+1} - 1\}$ be a CRS(2^{i+1}). Then, $\sum_{j=0}^{2^{i+1}-1} [R_j]_i = 2^i$ is always true, i.e., the sum of ith bits of R_j, for all $0 \leq j \leq 2^{i+1} - 1$, is equal to 2^i. Thus, it immediately implies that $\sum_{j \in \Psi(i,t)} [\rho(D(j))]_i = 2^i$ if

$\{\rho(D(j)) \mid j \in \Psi(i, t)\}$is a CRS($2^{i+1}$), $0 \le i \le n-1$ and $0 \le t \le 2^{n-i-1} - 1$.

(only if) From Theorem 7, if the permutation $\pi \in \Omega$, then $\{\rho(D(j)) \mid j \in \Psi(i, t)\}$ is a CRS(2^{i+1}), for any i and t. Thus, from the above fact, $\sum_{j \in \Psi(i,t)} [\rho(D(j))]_i = 2^i$, for any i and t. Since if $\pi \in \Omega$ then $\pi^{-1} \equiv D^{-1} \in \Omega_{-1}$, the equation $\sum_{j \in \Psi_R(i,t)} [D^{-1}(j)]_i = 2^i$ is also true.

(if) Note that actually $\sum_{j=0}^{2^{i+1}-1} [R_j]_k = 2^i$, for all $0 \le k \le i$, iff $\{R_j \mid 0 \le j \le 2^{i+1} - 1\}$ is a CRS(2^{i+1}). Thus, we need to prove the problem that if $\sum_{j \in \Psi(i,t)} [\rho(D(j))]_i = \sum_{j \in \Psi_R(i,t)} [D^{-1}(j)]_i = 2^i$, for any i and t, then it is sufficient to show that $\{\rho(D(j)) \mid j \in \Psi(i, t)\}$is a CRS($2^{i+1}$), for any i and t, which in turn implies that the permutation $\pi \in \Omega$.

According to Theorems 2 and 3, for any $0 \le i \le n-2$ and $0 \le t^*, t^{**} \le 2^{n-i-1} - 1$, there exist two sets $\Psi(i, t^*)$ and $\Psi(i, t^{**})$ such that $\Psi(i, t^*) \cap \Psi(i, t^{**}) = \emptyset$ and $\Psi(i, t^*) \cup \Psi(i, t^{**}) = \Psi(i+1, t)$. For $i = 0$, if the following conditions are true:

$\sum_{j \in \Psi(0,t^*)} [\rho(D(j))]_0 = 2^0$, (i.e., $\{\rho(D(j)) \mid j \in \Psi(0, t^*)\}$is a CRS(2),)

$\sum_{j \in \Psi(0,t^{**})} [\rho(D(j))]_0 = 2^0$, (i.e., $\{\rho(D(j)) \mid j \in \Psi(0, t^{**})\}$is a CRS(2),)

and $\sum_{j \in \Psi(1,t)} [\rho(D(j))]_1 = 2$,

then there are two possibilities:

(1) $\{\rho(D(j)) \mid j \in \Psi(1, t)\}$is a CRS($2^2$), i.e., $\{[\rho(D(j))]_{1,0} \mid j \in \Psi(1, t)\} = \{00, 01, 10, 11\}$.

(2) $\{[\rho(D(j))]_{1,0} \mid j \in \Psi(1, t)\} = \{00, 00, 11, 11\}$ or $\{01, 01, 10, 10\}$. For each case, there must exist some $\Phi[1, t']$ such that $\{[\rho(D(j))]_{1,0} \mid j \in \Psi(1, t')\} = \{00, 00, 11, 11\}$ or $\{01, 01, 10, 10\}$; otherwise there will be at least one $\Phi[1, t'']$ such that $\sum_{j \in \Psi(1,t'')} [\rho(D(j))]_1 \neq 2$. If this is true, then it will result in an odd number of 1's in routing tags which are used at stage b of at least one subnetwork $\Phi_R[\mathrm{b}, t''']$ where $0 \leq b \leq n-2$. That is, we have $\sum_{j \in \Psi(b,t''')} [D^{-1}(j)]_b \neq 2^b$ for at least one subnetwork $\Phi_R[\mathrm{b}, t''']$. Thus, it will always be detected that $\{\rho(D(j)) \mid j \in \Psi(1, t)\}$ is not a CRS(2^2).

By induction, we can show that if $\sum_{j \in \Psi(k,t)} [\rho(D(j))]_k = \sum_{j \in \Psi_R(k,t)} [D^{-1}(j)]_k = 2^k$, for all $0 \leq k \leq i$ and $0 \leq t \leq 2^{n-k-1}$, then, for each $0 \leq k \leq i$ and $0 \leq t \leq 2^{n-k-1}$, $\{\rho(D(j)) \mid j \in \Psi(k, t)\}$ is a CRS(2^{k+1}). Thus, if $\sum_{j \in \Psi(i,t)} [\rho(D(j))]_i = \sum_{j \in \Psi_R(i,t)} [D^{-1}(j)]_i = 2^i$, for any i and t, then $\{\rho(D(j)) \mid j \in \Psi(i, t)\}$is a CRS($2^{i+1}$), for any i and t. □

Note that Theorem 8 do not imply that if $\pi \in \Omega$ then $\pi \in \Omega^{-1}$, and vice versa. For example, let us consider the admissible permutation in Fig. 4.5 and check the subnetwork $\Phi[2, 1]$ where the corresponding set $\Psi(2, 1) = \{1, 3, 5, 7, 9, 11, 13, 15\}$. We have $\{[\rho(D(j))]_2 \mid j \in \Psi(2, 1)\} = \{[\rho(D(1))]_2, [\rho(D(3))]_2, [\rho(D(5))]_2, [\rho(D(7))]_2, [\rho(D(9))]_2, [\rho(D(11))]_2, [\rho(D(13))]_2, [\rho(D(15))]_2\} = \{[2]_2, [1]_2, [15]_2, [3]_2, [5]_2, [14]_2, [8]_2, [4]_2\} = \{0, 0, 1, 0, 1, 1, 0, 1\}$. Thus, $\sum_{j \in \Psi(2,1)} [\rho(D(j))]_2 = 2^2$. Moreover, $\Psi(2, 1) = \Psi(1, 1) \cup \Psi(1, 3)$ where $\Psi(1, 1) = \{1, 5, 9, 13\}$ and $\Psi(1, 1) = \{3, 7, 11, 15\}$. It also can be shown that $\sum_{j \in \Psi(1,1)} [\rho(D(j))]_1 = 2$ and $\sum_{j \in \Psi(1,3)} [\rho(D(j))]_1 = 2$. Similarly, $D^{-1} = (12, 13, 15, 10, 1, 6, 0,$

11, 3, 14, 9, 4, 7, 8, 2, 5). We can obtain that $\sum_{j \in \Psi_R(1,t)} [D^{-1}(j)]_1 = 2$.

Theorem 8 implies that for each permutable structure $\Phi[i, t]$, the work to determine the CRS property of $\{\rho(D(j)) \mid j \in \Psi(i, t)\}$, $\#(\Psi(i, t)) = 2^{i+1}$, in Theorem 7 can be replaced by two bit summation operations. That is, we sum the ith bits of all the destination tags $\rho(D(j))$, where $j \in \Psi(i, t)$, and sum the ith bits of all the destination tags $(D^{-1}(j))$, where $j \in \Psi_R(i, t)$. Then, we check whether or not both of the results are equal to 2^i. Both the admissibility conditions (Theorem 6) of the permutable substructures of an Omega network and its inverse network (an inverse Omega network) need to be satisfied. An algorithm to determine the admissibility of a permutation based on Theorem 8 is described as follows:

```
function ADMISSIBILITY-2 ($j \in \{0, ..., N-1\}$, $D$: permutation)
    for $i = 0$ to $n - 1$ do
        for $t = 0$ to $2^{n-i-1} - 1$ do
            if $\left[ \sum_{j \in \Psi(i,t)} [\rho(D(j))]_i = \sum_{j \in \Psi_R(i,t)} [(D^{-1}(j))]_i = 2^i \right] = false$
                then return false
    return true
```

Conclusions can be made for our work and previous ones in Theorem 5. Conditions in Theorem 5 are essentially the non-conflict criteria for any switching element(s). That is, no two paths of a permutation routing pass through the same input port of a switching element, i.e.,

$$([i]_{b-1:0}, [D(i)]_{n-1:b}) \neq ([j]_{b-1:0}, [D(j)]_{n-1:b}),$$

$$\text{for any } 0 \le i, j \le N-1,\ 1 \le b \le n-1.$$

This is a *one-dimension* viewpoint to understand what the admissible permutations of an Omega network are. On the other hand, our work exploits all the structures (subnetworks) which are relative to permutation routing behavior of an Omega network. Then, we develop the non-conflict criteria for these structures with the aid of the structure of its inverse Omega network which as mentioned above can be sufficiently represented by a very simple bit-summation condition. Thus, our work provides a *two−dimension* viewpoint to understand what the admissible permutations of an Omega network are. It is obvious that our method is simpler and easier than previous ones.

4.5. GENERAL MODEL

Generally speaking, there exists a class of topologically equivalent networks which are constructed by the BPC permutation connections and possess the unique-path and full-access properties. As we will see, even through this class of networks represents only a subset of Banyan networks, it provides more attractive communication aspects than other networks which are constructed by irregular connection patterns. For example, the BPC permutation connections for Omega networks are perfect shuffle permutations. Each network of this class has the similar routing behavior and thus the similar expression of transition sequences like Omega networks. This class of networks includes the six networks mentioned in [WuFe81] as special cases. The connection patterns used between stages of them are a specified set from $\{\beta_n\}$. Their transition sequence which represents any path connecting network input $s = (s_{n-1}, ..., s_1, s_0)$ to network output $D(s) = (d_{n-1}, ..., d_1, d_0)$ has the following properties.

(1) Each bit of the sources s (or its complement) will be permuted to the position of LSB in some s^i and then be replaced by a bit of the destination $D(s)$ (or its complement) in $D^i(s)$, $0 \leq i \leq n-1$. Therefore, there exist two permutation functions $O, I^{-1} \in \{\beta_n\}$ such that $O(s)$ corresponds to the order for bits of s to be permuted to the position of LSB (i.e. $[O(s)]_i$ is

the LSB in s^i) and $I^{-1}(D(s))$ corresponds to the order for bits of $D(s)$ to replace bits of s (i.e., $[I^{-1}(D(s))]_i$ replaces $[O(s)]_i$ in the LSB position of $D^i(s)$). The physical meaning of permutation function I is as follows: if the ith bit of a number $X = (x_{n-1}, x_{n-2}, ..., x_1, x_0)$ instead of bit $[I^{-1}(D(s))]_i$ replaces $[O(s)]_i$ in $D^i(s)$, then $I(X)$ represents the final destination where the source s will reach. These two BPC permutation functions O and I are referred to as characteristic functions of a network.

(2) Data packets from source s are routed from input port $[O(s)]_i$ to output port $[I^{-1}(D(s))]_i$ of a switching element at stage i and the address of this switching element is either $[s^i]_{n-1:1}$ or $[D^i(s)]_{n-1:1}$.

(3) The routing scheme of this class of networks can be described as follows. Let the symbol @ represent the exclusive-or operation. Bit $[I^{-1}(D(s))]_i$ is used as the routing tag for the switching element at stage i such that data packets are routed from input port $[O(s)]_i$ to output port $[I^{-1}(D(s))]_i$. Bit $[O(s)]_i$ @ $[I^{-1}(D(s))]_i$ is used to determine the state of the switching element at stage i if global routing is considered and no conflict occurs. That is, if $[O(s)]_i$ @ $[I^{-1}(D(s))]_i = 0$ then the switching element will be in a straight connection state (i.e., 0-state), else the switching element will be in an exchange connection (i.e., 1-state). After a data packet traverses stage i, bit $[O(s)]_i$ (i.e., the label of the input port from which this incoming data packet comes) is attached to this data packet in order to recover the information of the source address. We call this kind of routing scheme as the *source−preserved* and *destination−oriented* routing scheme. It is clear that the routing behavior of any network in this class can be uniquely characterized by functions O and I. Note that not all the networks with full access capability and unique-path property possess this kind of simple routing

scheme. In general, for a network with irregular connection patterns, the routing tag used at each stage is a function of both source input and destination output.

For example, in Fig. 4.6, a 16×16 4-stage network defined by a sequence of BPC permutation operations is shown. Let the characteristic functions of this network, O and I, be specified by vectors $\hat{O}$ and $\hat{I}$, respectively. Let the interconnection pattern P_i, $0 \le i \le n$, be specified by vector $\hat{P}_i$ and $\hat{P}_0 = (2, -1, -0, 3)$, $\hat{P}_1 = (2, 3, 0, -1)$, $\hat{P}_2 = (-0, 1, 3, -2)$, $\hat{P}_3 = (2, -0, 3, 1)$, $\hat{P}_4 = (1, 3, 0, 2)$. The transition sequence is:

$$s = (s_3, s_2, s_1, s_0) \qquad s^0 = (s_2, \bar{s}_1, \bar{s}_0, s_3)$$

$$D^0(s) = (s_2, \bar{s}_1, \bar{s}_0, d_2) \qquad s^1 = (\bar{s}_1, s_2, d_2, s_0)$$

$$D^1(s) = (\bar{s}_1, s_2, d_2, \bar{d}_3) \qquad s^2 = (d_3, d_2, \bar{s}_1, \bar{s}_2)$$

$$D^2(s) = (d_3, d_2, \bar{s}_1, \bar{d}_0) \qquad s^3 = (d_2, d_0, d_3, \bar{s}_1)$$

$$D^3(s) = (d_2, d_0, d_3, d_1) \qquad s^4 = (d_3, d_2, d_1, d_0) = D(s).$$

Thus, we have $\hat{O} = (-1, -2, 0, 3)$, $\hat{I} = (-1, 0, 3, -2)$, $\hat{I}^{-1} = (1, -0, -3, 2)$, $|\hat{O}| = (1, 2, 0, 3)$ and $|\hat{I}^{-1}| = (1, 0, 3, 2)$. For any path connecting a source s to a destination $D(s)$, bit d_2 is used as the routing tag at stage 0, bit $\bar{d}_3$ is used as the routing tag at stage 1, bit $\bar{d}_0$ is used as the routing tag at stage 2, and bit d_1 is used as the routing tag at stage 3. The states of switching elements from stage 0 to stage 3 are determined by $s_3 @ d_2$, $s_0 @ \bar{d}_3$, $\bar{s}_2 @ \bar{d}_0$,and $\bar{s}_1 @ d_1$, respectively. Hence, for the path connecting $s = 1$ to $D(s) = 4$, the routing tags are $[I^{-1}(D(s))] = (d_1, \bar{d}_0, \bar{d}_3, d_2) = (0, 1, 1, 1)$ and states are $(\bar{s}_1 @ d_1, \bar{s}_2 @ \bar{d}_0, s_0 @ \bar{d}_3, s_3 @ d_2) = (1, 0, 0, 1)$.

As a network in this class is specified by its two characteristic functions O and I, it can be shown that function O uniquely determines all the sets $\Psi(i, t)$, $0 \le i \le n - 1$ and $0 \le t \le$

$2^{n-i-1} - 1$, and $I^{-1}(D)$ is the actual permutation where computation should be performed to determine the admissibility of a given permutation D. That is, the permutation capability of this network is uniquely characterized by its two characteristic functions. Therefore, all our work done in previous sections can be easily extended to the general model by using characteristic functions. We summarize the generalization as the following theorems. In the following, let Γ be an n-stage network in the class specified by characteristic functions O and I.

THEOREM 9: Let Γ_R be the inverse network of Γ. Then, the characteristic functions of Γ_R are

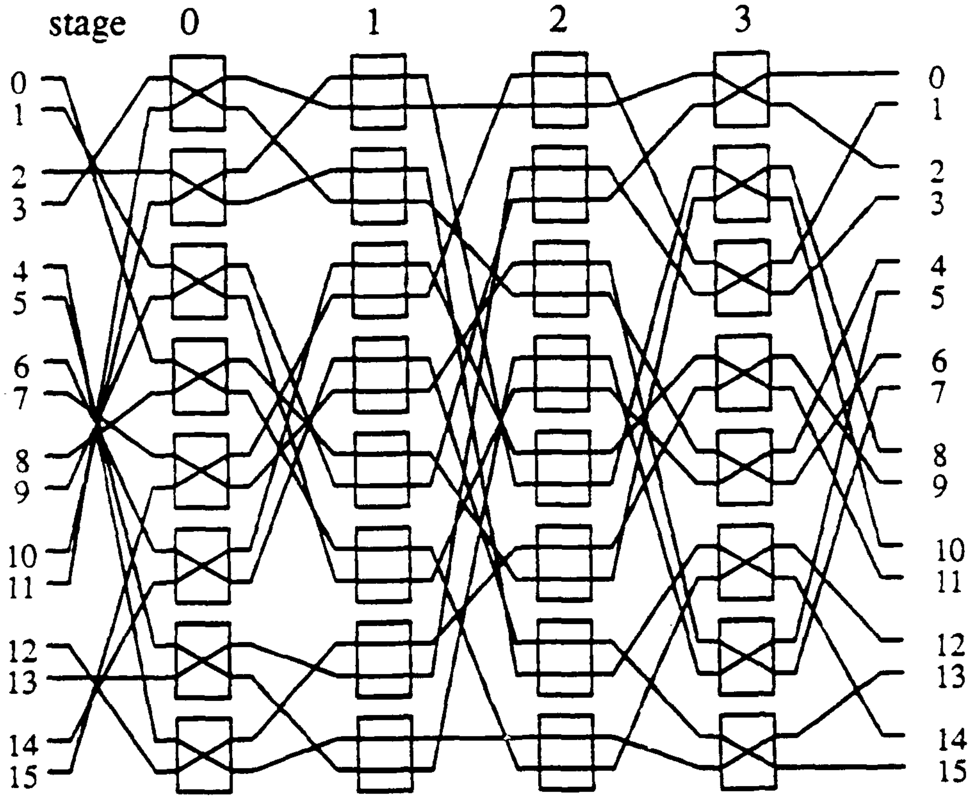
$$O_R = \rho \cdot I^{-1} \text{ and } I_R^{-1} = \rho \cdot O .$$

PROOF: For any transition sequence of Γ, we have a new explanation for Γ_R. Note that the ith stage of Γ_R becomes the $(n-i)$th stage of Γ. Thus, $\rho \cdot I^{-1}(D(s))$ corresponds to the order for bits of $D(s)$ to be permuted to the position of LSB (i.e., $[\rho \cdot I^{-1}(D(s))]_i$ is the LSB in $D^i(s)$) and $\rho \cdot O(s)$ corresponds to the order for bits of s to replace bits of $D(s)$ (i.e. $[\rho \cdot O(s)]_i$ replaces $[\rho \cdot I^{-1}(D(s))]_i$ in the LSB position of s^i). □

For example, let the interconnection pattern R_i, $0 \le i \le n$, be specified by vector $\hat{R}_i$ for the inverse network of the network in Fig. 4.6. Then, we have $\hat{R}_0 = \hat{P}_4^{-1} = (2, 0, 3, 1)$, $\hat{R}_1 = \hat{P}_3^{-1} = (1, 3, 0, -2)$, $\hat{R}_2 = \hat{P}_2^{-1} = (1, -0, 2, -3)$, $\hat{R}_3 = \hat{P}_1^{-1} = (2, 3, -0, 1)$, and $\hat{R}_4 = \hat{P}_0^{-1} = (0, 3, -2, -1)$. The transition sequence for connecting source input $D(s) = (d_3, d_2, d_1, d_0)$ to destination output $s = (s_3, s_2, s_1, s_0)$ is:

$$D(s) = (d_3, d_2, d_1, d_0) \qquad s^0 = (d_2, d_0, d_3, d_1)$$

$$D^0(D(s)) = (d_2, d_0, d_3, \bar{s}_1) \qquad s^1 = (d_3, d_2, \bar{s}_1, \bar{d}_0)$$



s	s	$D(s)$	$D(s)$	$I^{-1}(D(s))$
0	0 0 0 0	12	1 1 0 0	0 1 0 1
1	0 0 0 1	4	0 1 0 0	0 1 1 1
2	0 0 1 0	14	1 1 1 0	1 1 0 1
3	0 0 1 1	6	0 1 1 0	1 1 1 1
4	0 1 0 0	13	1 1 0 1	0 0 0 1
5	0 1 0 1	5	0 1 0 1	0 0 1 1
6	0 1 1 0	15	1 1 1 1	1 0 0 1
7	0 1 1 1	7	0 1 1 1	1 0 1 1
8	1 0 0 0	8	1 0 0 0	0 1 0 0
9	1 0 0 1	0	0 0 0 0	0 1 1 0
10	1 0 1 0	10	1 0 1 0	1 1 0 0
11	1 0 1 1	2	0 0 1 0	1 1 1 0
12	1 1 0 0	9	1 0 0 1	0 0 0 0
13	1 1 0 1	1	0 0 0 1	0 0 1 0
14	1 1 1 0	11	1 0 1 1	1 0 0 0
15	1 1 1 1	3	0 0 1 1	1 0 1 0

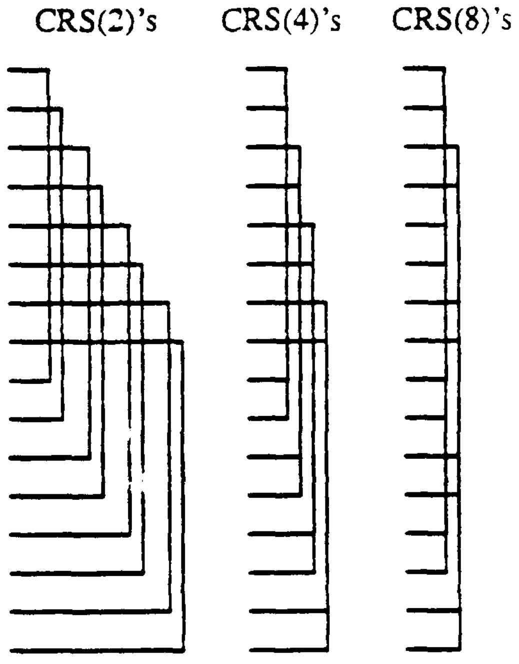


Fig. 4.6. A 16×16 network defined by the general model.

$$D^1(D(s)) = (d_3, d_2, \bar{s}_1, \bar{s}_2) \qquad s^2 = (\bar{s}_1, s_2, d_2, \bar{d}_3)$$

$$D^2(D(s)) = (\bar{s}_1, s_2, d_2, s_0) \qquad s^3 = (s_2, \bar{s}_1, \bar{s}_0, d_2)$$

$$D^3(D(s)) = (s_2, \bar{s}_1, \bar{s}_0, s_3) \qquad s^4 = (s_3, s_2, s_1, s_0) = s.$$

It is clear that $\hat{O}_R = \hat{\rho}\cdot(1, -0, -3, 2) = (2, -3, -0, 1)$ and $\hat{I}_R^{-1} = \hat{\rho}\cdot(-1, -2, 0, 3) = (3, 0, -2, -1)$.

THEOREM 10: A permutation π is admissible on Γ iff for each i and j, $0 \le i, j \le N - 1$ and $i \ne j$, either one of the following two conditions is true:

(1) $([I^{-1}(D(i))]_{b-1:0}, [O(i)]_{n-1:b}) \ne ([I^{-1}(D(j))]_{b-1:0}, [O(j)]_{n-1:b})$, for all $1 \le b \le n - 1$.

(2) $[I^{-1}(D(i))]_b = [O(i)]_b$ @ $f_b([I^{-1}(D(i))]_{b-1:0}, [O(i)]_{n-1:b+1})$, where @ is the exclusive-or operation and f_b is a Boolean function.

PROOF: This theorem is a generalized Theorem 5. By using the same criteria as Theorem 5, the above two conditions give the non-conflict criteria for any switching element(s) of Γ. That is, no two paths of the permutation π pass through the same input port of a switching element of Γ. □

THEOREM 11: By forcing all the switching elements of Γ at stage i to 0-state or 1-state, two disjoint $(n-1)$-stage subnetworks are formed such that in each subnetwork, the addresses of network inputs agree in bit $[|O|(s)]_i$ and the addresses of network outputs agree in bit $[|I^{-1}|(D(s))]_i$.

PROOF: The proof is similar to that of Theorem 1 except that we use $[O(s)]_i$ instead of s_{n-1-i}. That is, we have the following statement: forcing all switching elements at stage i to 0-state (1-state) is equivalent to forcing the LSB, $[O(s)]_i$, of s^i to be replaced by

$[I^{-1}(D(s))]_i = [O(s)]_i$ $(1 - [O(s)]_i)$ in the LSB position of $D^i(s)$. Thus, for 0-state case, the addresses of network inputs of each $(n-1)$-stage subnetwork agree in bit $[O(s)]_i = [I^{-1}(D(s))]_i$ and the addresses of network outputs agree in $[I^{-1}(D(s))]_i = [O(s)]_i$. Similarly, for 1-state case, the addresses of network inputs of each $(n-1)$-stage subnetwork agree in bit $[O(s)]_i = 1 - [I^{-1}(D(s))]_i$ and the addresses of network outputs agree in $[I^{-1}(D(s))]_i = 1 - [O(s)]_i$. □

For example, by forcing all the switching elements at stage 1 of the 16×16 4-stage network in Fig. 4.6, the addresses of network inputs of each 3-stage subnetwork agree in bit $[O(s)]_1 = s_0 = 1 - [I^{-1}(D(s))]_1 = \overline{(\overline{d})}_3 = d_3$ and the addresses of network outputs agree in $[I^{-1}(D(s))]_1 = \overline{d}_3 = 1 - [O(s)]_i = \overline{s}_0$. That is, each 3-stage subnetwork has network inputs $\{(c, c, c, s_0)\}$ and network outputs $\{(s_0, c, c, c)\}$. Thus, one of the two subnetworks has network inputs {0, 2, 4, 6, 8, 10, 12, 14} and network outputs {1, 2, 3, 4, 5, 6, 7}.

THEOREM 12: Let $\Phi[i, t]$, $0 \le i \le n - 1$ and $0 \le t \le 2^{n-i-1} - 1$, be a subnetwork produced by performing the same partitioning scheme (as mentioned in Section 4.3) on Γ. For each i, $0 \le i \le n - 1$, let $\{\Psi(i, u) \mid 0 \le u \le 2^{n-i-1} - 1\}$ be the partition on the network inputs $\{0, 1, ..., N-1\}$ corresponding to the set of subnetworks $\{\Phi[i, t] \mid 0 \le t \le 2^{n-i-1} - 1\}$. Then, $\Psi(i, u) = \{(v_{n-1}, ..., v_1, v_0)\}$ such that $v_l = c$, for all $l \neq j$, where $s_j = [|O|(s)]_k$, $i + 1 \le k \le n - 1$. Thus, $\#(c) = i + 1$ in $\Psi(i, u)$.

PROOF: The proof is similar to that of Theorem 2 and is based on Theorem 10. □

Theorem 12 shows that function $|O|$ uniquely determines all the n partitions $\{\Psi(i, u) \mid 0 \le u \le 2^{n-i-1} - 1\}$, $0 \le i \le n - 1$. For example, let us find the partition $\{\Psi(0, u) \mid 0 \le u \le 2^3 - 1\}$ for the 16×16 4-stage network in Fig. 4.6. Since $|O|(s) = (s_1, s_2, s_0, s_3)$, we have

$\{s_j\} = \{[|O|(s)]_k \mid 1 \le k \le 3\} = \{s_0, s_2, s_1\}$. Thus, $\Psi(0, u) = (c, v_2, v_1, v_0)$. In a similar way, we can find the other two partitions. Therefore, the three partitions $\{\Psi(i, t) \mid 0 \le t \le 2^{3-i} - 1\}$, $0 \le i \le 2$, corresponding to the three sets of subnetworks $\{\Phi[i, t] \mid 0 \le t \le 2^{3-i} - 1\}$, $0 \le i \le 2$, are

$\{\Psi(0, u)\}$

$$= \{\{(c, 0, 0, 0)\}, \{(c, 0, 0, 1)\}, \{(c, 0, 1, 0)\}, \{(c, 0, 1, 1)\}, \{(c, 1, 0, 0)\}, \{(c, 1, 0, 1)\}, \{(c, 1, 1, 0)\}, \{(c, 1, 1, 1)\}\}$$

$$= \{\{0, 8\}, \{1, 9\}, \{2, 10\}, \{3, 11\}, \{4, 12\}, \{5, 13\}, \{6, 14\}, \{7, 15\}\},$$

$\{\Psi(1, u)\}$

$$= \{\{(c, 0, 0, c)\}, \{(c, 0, 1, c)\}, \{(c, 1, 0, c)\}, \{(c, 1, 1, c)\}\}$$

$$= \{\{0, 1, 8, 9\}, \{2, 3, 10, 11\}, \{4, 5, 12, 13\}, \{6, 7, 14, 15\}\},$$

$\{\Psi(2, u)\}$

$$= \{\{(c, c, 0, c)\}, \{(c, c, 1, c)\}\}$$

$$= \{\{0, 1, 4, 5, 8, 9, 12, 13\}, \{2, 3, 6, 7, 10, 11, 14, 15\}\}.$$

THEOREM 13: The permutation π is admissible on Γ iff $\{I^{-1}(D(j)) \mid j \in \Psi(i, t)\}$ is CRS(2^{i+1}) or $\{|I^{-1}|(D(j)) \mid j \in \Psi(i, t)\}$ is CRS(2^{i+1}), for all $0 \le i \le n - 1$ and $0 \le t \le 2^{n-i-1} - 1$.

PROOF: This proof is similar to those of Theorems 6 and 7. Note that any permutation function $\beta \in \{\beta_n$ is closed on domain $\{0, 1, ..., N - 1\}$, i.e., $\{\beta(i) \mid 0 \le i \le N - 1\} = \{0, 1, ..., N - 1\}$. Thus, it is clear that $\{I^{-1}(D(j)) \mid j \in \Psi(i, t)\}$ is CRS(2^{i+1}) iff $\{|I^{-1}|(D(j)) \mid j \in \Psi(i, t)\}$ is CRS(2^{i+1}). Therefore, $\{I^{-1}(D(j)) \mid j \in \Psi(i, i)\}$ is CRS(2^{i+1}) gives the necessary and sufficient condition for non-conflict at any switching element(s) of subnetwork $\Phi[i,$

t]. □

THEOREM 14: The permutation π is admissible on Γ iff $\sum_{j\in \Psi(i,t)} [I^{-1}(D(j))]_i = \sum_{j\in \Psi_R(i,t)} [\rho\cdot O(D^{-1}(j))]_i = 2^i$ or $\sum_{j\in \Psi(i,t)} [|I^{-1}|(D(j))]_i = \sum_{j\in \Psi_R(i,t)} [\rho\cdot|O|(D^{-1}(j))]_i\ 2^i$, for all $0 \le i \le n-1$ and $0 \le t \le 2^{n-i-1}-1$.

PROOF: This proof is similar to that of Theorem 8. It can be proved that the condition $\sum_{j\in \Psi(i,t)} [I^{-1}(D(j))]_i = \sum_{j\in \Psi_R(i,t)} [I_R^{-1}(D^{-1}(j))]_i = \sum_{j\in \Psi_R(i,t)} [\rho\cdot O(D^{-1}(j))]_i = 2^i$, for any i and t, is sufficient to derive that $\{I^{-1}(D(j)) \mid j \in \Psi(i,t)\}$is a CRS($2^{i+1}$), for any i and t, which in turn implies that the permutation π is admissible on Γ. □

For example, in Fig. 4.6, we also show why a permutation (12, 4, 14, 6, 13, 5, 15, 7, 8, 0, 10, 2, 9, 1, 11, 3) is admissible on this network.

THEOREM 15: Let Γ_x and Γ_y be two networks specified by characteristic functions O_x, I_x and O_y, I_y, respectively. Γ_x and Γ_y are in a subclass of equivalent networks with the same set of admissible permutations iff $|O_x| = |O_y|$ and $|I_x| = |I_y|$.

PROOF: This proof is based on Theorems 12, 13 and 14. According to Theorem 12, function O uniquely determines all the n partitions $\{\Psi(i,t) \mid 0 \le t \le 2^{n-i-1}-1\}$, $0 \le i \le n-1$. And according to Theorems 13 and 14, for any permutation D, the results of bit-summation and comparison operations performed on set $\{I^{-1}(D(j)) \mid 0 \le j \le N-1\}$ are the same as those on $\{|I^{-1}|(D(j)) \mid 0 \le j \le N-1\}$. Thus, any two networks with the same absolute characteristic functions have the same set of admissible permutations. □

Theorem 15 provides a direct view of equivalence between networks by the set of admissible permutations. Some authors [Par80] [Agr83] denoted it as *functional equivalence*. That is, if two networks can realize the same set of permutations, then they are functionally equivalent. According to Theorem 15, for any pair of functions $O, I \in \{\beta_n\}$, if $|O| = O$ and $|I| = I$, there exists a subclass of functionally equivalent networks with the same set of admissible permutations which are characterized by O and I. Since there are $n! = \log_2 N!$ of such function O's or I's, it is easy to show that the whole topologically equivalent class of networks can be partitioned into $(\log_2 N)^2$ disjoint subclasses. In any subclass, each network is not only a different drawing of another network but also realizes the same set of permutations.

For example, in Table 4.1, the general form of the characteristic functions and the set of partitions of several famous networks are shown. From Table 4.1, we obtain the following facts. The Baseline and inverse Baseline network have the same set of admissible permutations. The Omega and inverse Indirect Binary Cube network have the same set of admissible permutations. The Indirect Binary Cube and inverse Omega network have the same set of admissible permutations.

In Parker's work [Par80], the functional equivalence of three networks (i.e., the inverse Omega, Indirect Binary Cube and R-network) are proved. Identity relations between several specific permutation functions are used to transform a network to another one. Even though, conceptually, the method can be generalized (which in our opinion will be very complicated) to prove the functional equivalence of other networks, it restricts our view to a one-dimension solution as that in condition (2) of Theorem 5 to outline what the permutations which a network can realize really look like. It is clear that our method provides a two-dimension solution by simple bit relations and a more direct insight than that in [Par80] to describe the

Table 4.1. The general forms of the characteristic functions of several famous networks.

Networks	$\hat{O}$	$\hat{I}^{-1}$	$\Psi(i, u) = \{(v_{n-1}, \ldots, v_1, v_0)\}$
Delta network	$(1, 2, \ldots, n-1, 0)$	$(0, 1, \ldots, n-2, n-1)$	$\{(c, \ldots, c, v_{n-i-1}, \ldots, v_1, c)\}$
inverse Delta network	$(n-1, n-2, \ldots, 1, 0)$	$(0, n-1, \ldots, 2, 1)$	$\{(v_{n-1}, \ldots, v_{i+1}, c, \ldots, c)\}$
Omega network	$(0, 1, \ldots, n-2, n-1)$	$(0, 1, \ldots, n-2, n-1)$	$\{(c, \ldots, c, v_{n-i-2}, \ldots, v_0)\}$
inverse Omega network	$(n-1, n-2, \ldots, 1, 0)$	$(n-1, n-2 \ldots, 1, 0)$	$\{(v_{n-1}, \ldots, v_{i+1}, c, \ldots, c)\}$
Baseline network	$(n-1, n-2, \ldots, 1, 0)$	$(0, 1, \ldots, n-2, n-1)$	$\{(v_{n-1}, \ldots, v_{i+1}, c, \ldots, c)\}$
inverse Baseline network	$(n-1, n-2, \ldots, 1, 0)$	$(0, 1, \ldots, n-2, n-1)$	$\{(v_{n-1}, \ldots, v_{i+1}, c, \ldots, c)\}$
Indirect Binary Cube network	$(n-1, n-2, \ldots, 1, 0)$	$(n-1, n-2, \ldots, 1, 0)$	$\{(v_{n-1}, \ldots, v_{i+1}, c, \ldots, c)\}$
inverse Indirect Binary Cube network	$(0, 1, \ldots, n-2, n-1)$	$(0, 1, \ldots, n-2, n-1)$	$\{(c, \ldots, c, v_{n-i-2}, \ldots, v_0)\}$

meaning of functional equivalence.

4.6. SUMMARY

In this chapter, by employing a proper partitioning scheme, the properties of a number of permutable substructures (subnetworks) on an Omega network are studied. These substructures are associated with some specific partitions on the network inputs and can be used to characterize admissible permutations of an Omega network. Based on the understanding of these substructures, the permutation capability of Omega networks is characterized by either using the residue classes or bit relations of destination tags. We propose an algorithm to determine the admissibility of a permutation on an Omega network which has a time complexity $O(N)$, where N is the number of inputs/outputs of the network. Finally, we generalize the same methodology used on Omega networks to a class of topologically equivalent networks defined by BPC permutations in which each network can be specified by two characteristic functions.

CHAPTER 5

A FAULT-TOLERANT RECONFIGURATION SCHEME FOR MULTIPROCESSORS

5.1. INTRODUCTION

One of the most cost-effective ways for interconnecting a very large number of processors to form a *general-purpose* multiprocessor system is to employ a Multistage Interconnection Network (MIN) [WuFe81] [Par80] [Pea77]. In such a system, the MIN which is a critical component provides a full access communication between processors However, physical failures in a MIN can cause severe degradation in the system performance, unless efficient methods are provided to handle them.

Various issues concerning the analysis of fault tolerance capability and reliability of multiprocessor systems with MINs have been studied in [GaMa88] [DaBh85]. In one of these methods, the failure of a switching element in the network causes the removal of a number of processors such that the system can operate in a degraded mode in which the full access property can be maintained among the remaining processors. However, this strategy results in an enormous waste of computational resources. As a MIN is used for interprocessor connection, an alternate strategy to minimize the loss of computational resources is to allow the communication with multiple passes through the faulty network by using the remaining fault-free paths. A multiprocessor system with a faulty network is said to possess the *dynamic full access* (DFA) property if each processor in the system can communicate with any other processor in the system in a finite number of passes through the faulty network, by routing the data through proper intermediate processors if necessary [ShHa84] [VaRa89]. This strategy results in a reconfigured system which can operate in a gracefully degraded mode at the expense of routing overhead, the increased latency and the additional blocking due to the loss of communication paths. As the studies in [ShHa84] [VaRa89] [AgLe85] have shown, the general problem of determining the DFA property of a faulty network is as hard as a transitive closure problem. No general necessary and sufficient conditions have been found yet to determine the DFA property based on the distribution of faulty components on the network.

A successful survival of a multiprocessor system in the presence of network failures requires solutions of the following problems.

(1) A fast and effective fault testing algorithm to detect failures of the network.

(2) A multi-fault diagnosis algorithm to locate all the faults.

(3) A real-time reconfiguration scheme to prevent the waste of additional computational effort.

Several studies of fault testing and diagnosis algorithms can be found in [Agr82] [ThNe83] [WuFe79] [NaSo80] [FuAb83] [Agr80] [FaPr81]. In this chapter, we address only problem (3). We assume that the information of locations of all the faulty components in the network is available. Central to the design of such a reconfiguration scheme is the utilization of this given information to reconfigure the system into a single (sub)system or several subsystems with DFA property such that the original routing scheme can be preserved in each subsystem. The fault-tolerant reconfiguration scheme to be presented is suitable for the on-line and real-time applications. The scheme is simple, efficient, and applicable to all the networks discussed in the literature. A special network topology, the Omega network [Law75], is used as the example network in this chapter. Several important problems which have not been previously considered are addressed in our work and discussed in reference to an intergrated model. Those which distinguish this chapter from previous work [ShHa84] [VaRa89] [AgLe85] are summarized as follows:

(1) In many faulty situations, some processors might be completely isolated from other processors (i.e., no fault-free paths exist between them and other processors). If this information is not known, the data communication to/from these processors will block other fault-free communication paths and significantly degrade system performance. Therefore, it is extremely important for the system to obtain this information in order to disable these processors and obtain a better communication load control. In this chapter, this information, which is missing in previous work, is obtained.

(2) In [ShHa84] [VaRa89] [AgLe85], the authors were only interested in determining the sufficient conditions for a faulty network to possess the DFA property. In this chapter, we will show that even if the original system does not have the DFA property due to faults in the network, the surviving system obtained after disabling those processors defined in (1) may have the DFA property. Since there exist many possible multipass communication paths between surviving processors (those processors in the surviving system), an efficient way to achieve low latency communication is by utilizing of

shortest-path routes between these processors. While, in [VaRa89], Varma and Raghavendra mentioned that this is a very important issue, they did not actually show how to do it. In this chapter, a shortest-path fault-tolerant routing scheme is developed such that by routing through proper intermediate processors a processor can access another processor with a minimal number of passes through the faulty network.

(3) Since an acknowledge signal and bidirectional data communication are always required, it is necessary that bidirectional communication paths exist between any two processors. However, in some situations, there may exist only unidirectional communication paths between two processors. Such situations, as we will show, are due to the non-DFA property of the surviving system. The use of such unidirectional communication paths will cause a a deadlock because no possible acknowledge signals will be received by the source processor. Therefore, the utilization of shortest-path routes alone may not be sufficient to survive a system. An algorithm to prevent deadlocks must also be employed. In this chapter, such an algorithm is proposed which gives the solution in a way that the surviving system is partitioned into several surviving subsystems and each subsystem is a maximal subset of processors which possesses the DFA property.

In summary, the fault-tolerant reconfiguration scheme to be presented provides a flexible reconfigurable environment for a multiprocessor system with a faulty network. Under such an environment, the communication of the surviving system is operated by using the information of shortest-path routes. The rest of this chapter is organized as follows. In Section 5.2, the system and fault models are presented. In Section 5.3, the fault-tolerant reconfiguration scheme is presented. This scheme contains five parts: routing behavior of Omega networks under faults, communication capability for the first pass under faults, construction of the surviving system, construction of shortest-path routing tables, and reconfiguration of the surviving system. In Section 5.4, the time complexity of our scheme is analyzed. Finally, Section 5.5 gives the summary of this chapter.

5.2. SYSTEM AND FAULT MODELS

5.2.A. System Model

In this chapter, without loss of generality, we limit our discussion to an N-processor system interconnected by an Omega network [Law75] built with 2×2 switching elements (see Fig. 5.1). Such a multiprocessor system is connected to and monitored by a front-end host computer. The overall system configuration can be either SIMD or MIMD structure depending on requirements of specific applications. An $N\times N$ Omega network with N network inputs and N network outputs consists of $n = \log_2 N$ stages of 2×2 switching elements. Each stage consists of $N/2$ switching elements and the interconnection pattern between stages is the perfect shuffle permutation. Each switching element allows point-to-point or broadcast communication from its input ports to output ports if no conflict occurs. For example, in Fig. 5.2, a 16-processor multiprocessor system connected by a 16×16 Omega network is shown. The following conventional notations are used throughout this chapter. The stages of the network are numbered from 0 through $n - 1$ from left to right and the input/output ports (including network inputs/outputs) of switching elements at each stage are numbered from 0 through $N - 1$ from top to bottom. The communication links between stages are numbered according to the order of input ports of the stage to which these links are connected. The communication links before stage 0 and after stage $n - 1$ are considered as pseudo links since they are connected to output ports and input ports of processors, respectively. For example, the labels of links connected to the input ports of stage i are shown in Fig. 5.2. Stage 0 is sometimes referred to as the *input stage* and stage $n-1$ as the *output stage*. Each network input and output are connected to the output and input ports of a processor with the same address. The address of a label L is represented by its binary form $L = (l_{n-1}, ..., l_1, l_0)$, where bit l_0 is the least significant bit (LSB). Thus, a switching element in the network is represented by an ordered pair (r, e), where r $(0 \le r \le n-1)$ is the stage at which the element is located and $e = (e_{n-1}, ..., e_1)$ $(0 \le e \le N/2 - 1)$ is the element address at that stage such that the two input/output ports of e have a common address label equal to $(e_{n-1}, ..., e_1, c)$, $c = 0$ or 1. A communication link is represented by an ordered pair $[t, h]$, where $h = (h_{n-1}, ..., h_1, h_0)$ $(0 \le h \le N-1)$ is the address of this link and t $(1 \le t \le n-1)$ is the stage at which the input port of a switching element $(h_{n-1}, ..., h_1)$ is connected to this link. Also the sets $\{[0, h]\}$ and $\{[n, h]\}$ are used to represent those pseudo links before stage 0 and after stage $n-1$. A set of labels with similar address representations can be denoted by a common address label. For example, $(l_{n-1}, l_{n-2}, ..., l_i, c, ..., c)$ where $\#(c) = i$ (i.e., the total number of c's is i) represents those 2^i labels with the same first $n-i$ bits in their addresses. The notation $L[a:b]$, $a \ge b$, is used to represent a segment of the address L from bit l_a to bit l_b i.e., $L[a:b] = (l_a, l_{a-1}, ..., l_{b+1}, l_b)$.

5.2.B. Fault Model

The fault model we consider is one that both the switching elements and communication links may fail. The faulty components (switching elements and/or communication links) are treated as unusable and no connection can be routed through them. Thus, a faulty set F on an Omega network is defined as a set of faulty components under consideration. Some standard definitions have been used in several previous studies [ShHa84] [VaRa89]. We quote

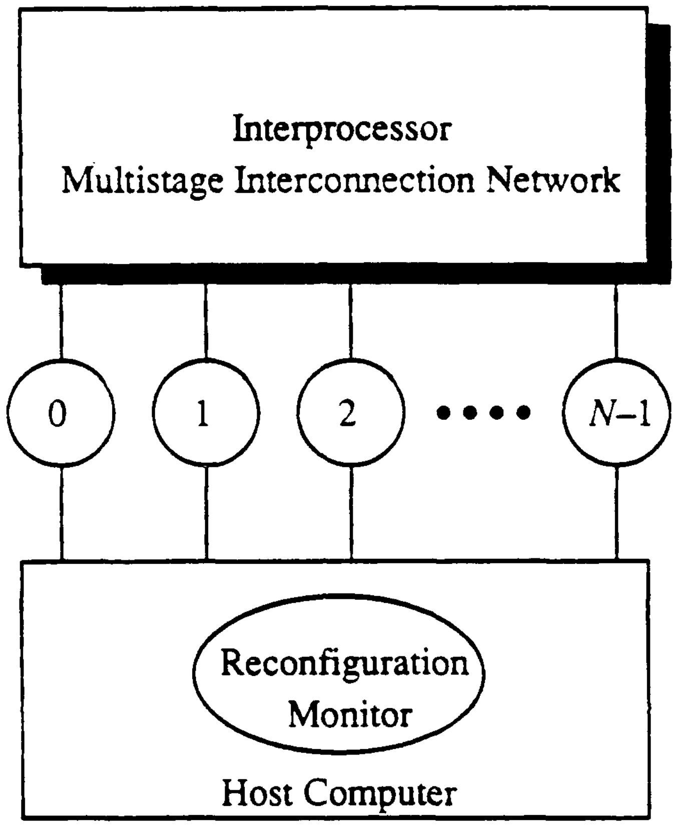


Fig. 5.1. System model.

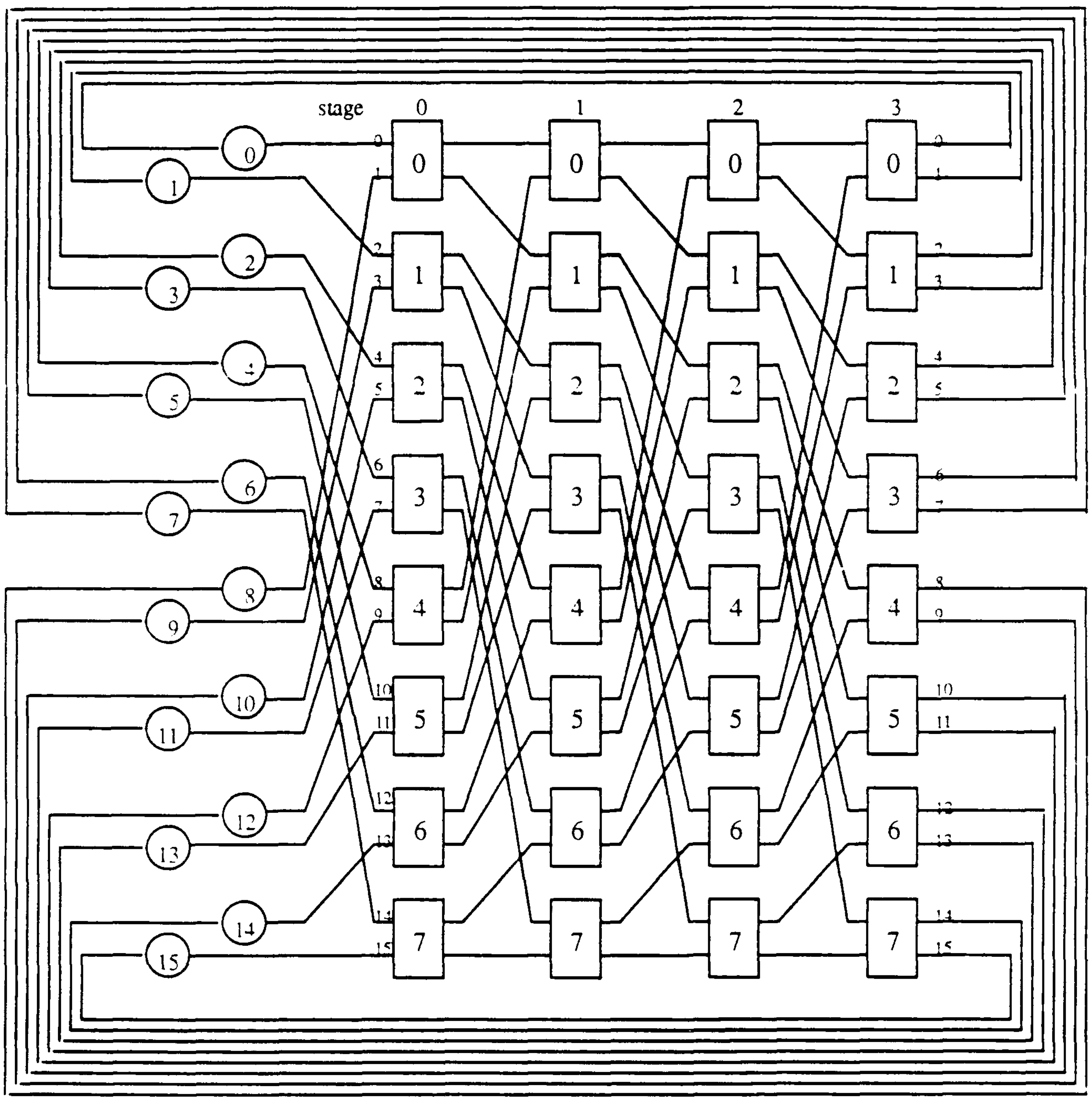


Fig. 5.2. A 16–processor multiprocessor system interconnected by a 16 × 16Omega network.

them here and extend their definitions for our convenience.

DEFINITION 1: A fault set F in an Omega network is *critical* with respect to a subsystem (system) iff it destroys the property of dynamic full access of this subsystem (system). □

Here, the definition of the DFA property is no longer restricted on the original N-processor system but on any subsystem composed of a subset of the N processors. Hence, if F is noncritical, data packets from any processor (or network input) can be routed to any other processor (or network output) in a finite number of passes through the faulty network.

DEFINITION 2: Let π be a permutation passable by an Omega network. The set of faulty paths $C_{F,\pi}$ of the permutation π under the fault set F is the set of communication paths that pass through some components in F when the system tries to realize π on the network. □

For example, let I be the identity permutation. The fault set $F = \{(1,0), (2,0), (2,1), [1,1], [1,8]\}$ in the network of Fig. 5.2 will affect those data packets which pass through the paths $0 \rightarrow 0$, $2 \rightarrow 2$, $4 \rightarrow 4$, and $6 \rightarrow 6$ of the identity permutation.

DEFINITION 3: Two fault sets F and F' are *equivalent* iff $C_{F,\pi} = C_{F',\pi}$ for all possible π. The notation $F \equiv F'$ is used to denote that F and F' are equivalent. □

For example, the fault set $F' = \{(1,0), (1,4), (2,0), (2,1), [1,0], [1,1], [1,8]\}$ is equivalent to $F = \{(1,0), (2,0), (2,1), [1,1], [1,8]\}$ in the network of Fig. 5.2 because F and F' affect the same set of communication paths of each passable permutation.

DEFINITION 4: The *maximal fault set* $F_{\max}$ corresponding to a fault set F in an Omega network is the set of maximum size that is equivalent to F. That is to say, $F_{\max} \equiv F$ and $F_{\max} \supseteq F'$, for all $F' \equiv F$. □

5.3. FAULT-TOLERANT RECONFIGURATION

In this section, we study a fault-tolerant reconfiguration scheme for an N-processor system with multiple faults on its Omega network. Such a scheme provides the system a flexible reconfigurable environment no matter whether or not the fault set under consideration is critical. A single surviving system or several surviving subsystems are formed by performing this scheme such that deadlocks can be avoided. This single surviving system may be composed of the original N processors or only a subset of them. A shortest-path routing table for each processor is obtained from which a processor can always know the minimal number of passes and proper intermediate processors to access other processors in the same surviving system (subsystem). The idea of the fault-tolerant reconfiguration is described as follows. In Fig. 5.1, imagine that there is a machinism in the host computer, named as *reconfiguration monitor*, which can process the fault-tolerant reconfiguration scheme. Once the locations of faulty components have been known, the reconfiguration monitor then processes the following procedures.

(1) Obtain the communication capability of each processor for the first pass through the faulty network. Some processors may be considered as unusable due to the complete destruction of their communication capabilities. Conceptually, a single surviving system will be produced if we remove these unusable processors.

(2) The communication capabilities of all the processors is sent back from the reconfiguration monitor to each processor. A shortest-path routing algorithm is performed in each processor to find all the possible shortest routes to other processors. Eventually, a shortest-path routing table is produced for each processor. By using this algorithm, the proper intermediate processors and the minimal number of passes through

the faulty network for a processor to access another processor are obtained.

(3) Under some situations, a reconfiguration algorithm must be employed to avoid the implicit danger of deadlocks. These situations are due to the criticality of the fault set with respect to the surviving system. According to this reconfiguration algorithm, the surviving system is partitioned into several subsystems. Each subsystem possesses the DFA property. The partitioning of the surviving system is implicitly equivalent to sacrificing some usable components which only help establishing unidirectional multi-pass communication paths. However, we do not have to know the actual locations of these usable components during the partitioning.

We start our discussion from basic properties of routing behavior of Omega networks under faults.

5.3.A. Routing Behavior of Omega Networks under Faults

For an n-stage Omega network, due to its regular structure, any paths traversing a switching element (i, e) at stage i can be expressed by the following two transition sequences. These transition sequences also indicate which network inputs S's and outputs D's are connected through e.

Backward:

$$E = (e_{n-1}, e_{n-2}, \ldots, e_1, c)$$

$$E^i = (e_{n-1}, e_{n-2}, \ldots, e_1, c)$$

$$D^{i-1} = (c, e_{n-1}, \ldots, e_2, e_1)$$

$$E^{i-1} = (c, e_{n-1}, \ldots, e_2, c)$$

......

$$D^0 = (c, \ldots, c, e_{n-1}, \ldots, e_{i+1}, c)$$

$$S = (c, \ldots, c, e_{n-1}, \ldots, e_{i+2}, e_{i+1})$$

Forward:

$$E = (e_{n-1}, e_{n-2}, \ldots, e_1, c)$$

$$E^{i+1} = (e_{n-2}, e_{n-3}, \ldots, e_1, c, e_{n-1})$$

$$D^{i+1} = (e_{n-2}, e_{n-3}, \ldots, e_1, c, c)$$

......

$$E^{n-1} = (e_i\ e_{i-1}, \ldots, e_1, c, \ldots, c, e_{i+1})$$

$$D^{n-1} = (e_i\ e_{i-1}, \ldots, e_1, c, \ldots, c, c)$$

$$D = (e_i\ e_{i-1}, \ldots, e_1, c, \ldots, c, c)$$

In these transition sequence, each E^j, $0 \le j \le n-1$, represents the address of the input port through which a path traverses stage j and each D^j, $0 \le j \le n-1$, the output port through which the path traverses stage j. Obviously, $E^j[n-1:1] = D^j[n-1:1]$ is the address of the switching element through which a path traverses stage j and each one of such paths passes through the switching element e at stage i (i.e., when $j = i$). The switching element e at stage i can be viewed as the common root of two communication binary trees. One is the backward $(i+1)$-level tree with the address label of its leaves (switching elements at input stage) equal to $(c, \ldots, c, e_{n-1}, e_{n-2}, \ldots, e_{i+1})$, $\#(c) = i$, and the address label of network inputs connected to leaves equal to $S = (c, \ldots, c, e_{n-1}, e_{n-2}, \ldots, e_{i+1})$, $\#(c) = i+1$. The other one is the forward $(n-i)$-level tree with the address label of its leaves (switching elements at output stage) equal to $(e_i, e_{i-1}, ..., e_1, c, ..., c)$, $\#(c) = n - i - 1$, and the address label of network outputs connected to leaves equal to $D = (e_i, e_{i-1}, ..., e_1, c, ..., c)$, $\#(c) = n - i$. Thus, totally 2^{i+1} network inputs and 2^{n-i} network outputs are connected through e. If e is faulty, clearly, the communication from these 2^{i+1} network inputs to those 2^{n-i} network outputs will be destroyed. Particularly, when $i = 0$ ($i = n - 1$), these above two binary trees reduce to one in which the root e is rooted at input (output) stage and the root e is connected to two network inputs $S = (c, e_{n-1}, e_{n-2}, ..., e_1)$ (two network outputs, $D = (e_{n-1}, e_{n-2}, ..., e_1, c)$ and all the network outputs (inputs). It is obvious that as the necessary condition for a fault set F to be noncritical with respect to the original N-processor system, F cannot contain any switching elements from the input or output stages; otherwise communication trees rooted at these switching elements are completely destroyed. If that happens, processors connected to faulty switching elements at input or output stages will no longer be used.

It also can be observed that the pair of switching elements $\{(i, w), (i, w + 2^{n-2})\}$ at stage i is connected to only one pair of switching elements $\{(i + 1, 2w), (i + 1, 2w + 1)\}$ at the next stage, where $0 \le w \le N/4 - 1$. This is referred to as the *buddy property* in [Agr83]. If both elements of a buddy pair $\{(i, w), (i, w + 2^{n-2})\}$ are in a fault set F, then F can be expanded to include elements $\{(i + 1, 2w), (i + 1, 2w + 1)\}$ without affecting any additional communication paths. Similarly, if both elements of $\{(i + 1, 2w), (i + 1, 2w + 1)\}$ are in F, then $\{(i, w), (i, w + 2^{n-2})\}$ can be included in F.

A similar argument can be made about a communication link $[i, h]$. The communication link $[i, h]$ between stage $i-1$ and stage i, $1 \le i \le n - 1$, can be viewed as the common root of two communication binary trees. One is the backward $(i+1)$-level tree with the address label of its leaves (i.e., processors connected to input stage) equal to $S = (c, ..., c, h_{n-1}, h_{n-2}, ..., h_i)$, $\#(c) = i$. The other one is the forward $(n-i+1)$-level tree with the address label of its leaves (i.e., processors connected to output stage) equal to $(h_{i-1}, h_{i-2}, ..., h_0, c, ..., c)$, $\#(c) = n - i$. Thus, totally 2^i network inputs and 2^{n-i} network outputs are connected through $[i, h]$. If $[i, h]$ is faulty, clearly, the communication from these 2^i network inputs to those 2^{n-i} network outputs will be destroyed. One thing which differs faulty communication links from faulty switching elements is that no failure of any single link will completely destroy the communication capability of any processor, even if the faulty link comes from $\{[1, h]\}$ and $\{[n - 1, h]\}$. However, since the pair of links $\{[i, 2e], [i, 2e + 1]\}$ is connected to the pair of input ports of switching element e at stage i and the pair of links $\{[i, e \textit{ div } 2^{n-2} + 4 \cdot (e \textit{ mod } 2^{n-2})], [i, e \textit{ div } 2^{n-2} + 4 \cdot (e \textit{ mod } 2^{n-2}) + 2]\}$, connected to the pair of output ports of switching element e at stage $i-1$, $0 \le e \le N/2 - 1$, the failure of such a pair of links is equivalent to the failure of a switching element. Thus, a fault set which contains such a pair of faulty links can include a switching element to which the pair of faulty links is connected in order to form an equivalent fault set.

For example, in Fig. 5.3, a fault set $F = \{(1,1), (1,2), (1,5), (2,0), (2,5), [1,1], [1,3]\}$ is shown on a 16×16 Omega network. The buddy pair of faulty switching elements $\{(1,1), (1,5)\}$ at stage 1, is connected to the buddy pair $\{(2,2), (2,3)\}$ at stage 2. Therefore, the fault set F can include $\{(2,2), (2,3)\}$ to form an equivalent fault set. Similarly, the pair of faulty links $\{[1,1], [1,3]\}$ is connected to the switching element $\{(0,4)\}$. Therefore, the fault set F can include $\{(0,4)\}$ as well to form another equivalent fault set.

5.3.B. Communication Capability for the First Pass under Faults

After the faulty components of a fault set have been located, an identical procedure is performed in each processor to obtain the communication capability of the N-processor system for the first pass through the faulty network. The following algorithm is a procedure to find all the accessible processors of processor i, $0 \le i \le N-1$, for the first pass through the faulty network. The notations ζ_L and ζ_{SW} represent a faulty link and a faulty switching element in a fault set F, respectively. Finally, the set Z_i contains all the accessible processors of processor i.

Algorithm 1:

```
{ Find all the accessible processors of processor $i$ for the first pass }
**procedure** Accessibility ($i$: processor; $F$ : a fault set)
{ let the binary representation of $i = (i_{n-1}, i_{n-2}, ..., i_0)$ }
    $k \leftarrow 0$
    $Z_i \leftarrow \{0, 1, ..., N-1\}$   { initially, $Z_i$ contains all the $N$ processors }
    **for** each $\zeta_L, \zeta_{SW} \in F$ **do**
        $mark[\zeta_L] \leftarrow True$
        $mark[\zeta_{SW}] \leftarrow True$
    { scan all the faulty components stage by stage }
    **while** $k \ne n-1$ **and** $Z_i \ne \phi$ **do**
        { delete those unaccessible processors due to the faulty links }
        **if** $k \ne 0$ **then**
            **for** each $\zeta_L = [k, (i_{n-k-2}, i_{n-k-3}, ..., i_0, x_{k-1}, x_{k-2}, ..., x_0, i_{n-k-1})] \in F$
            **do**
                **if** $mark[\zeta_L] \ne False$ **then**
                    $Z_i \leftarrow Z_i / \{(x_{k-1}, x_{k-2}, ..., x_0, c, ..., c) \mid \#(c) = n-k\}$
                    **for** each
                    $\zeta_L = [l, (i_{n-l-2}, i_{n-l-3}, ..., i_0, x_{k-1}, x_{k-2}, ..., x_0, y_{l-k-1}, y_{l-k-2}, ...,$
                    $y_0, i_{n-l-1})] \in F$,
                    $\zeta_{SW} = (j, (i_{n-j-2}, i_{n-j-3}, ..., i_0, x_{k-1}, x_{k-2}, ..., x_0, y_{j-k-1}, y_{j-k-2},$
                    $..., y_0)) \in F$,
```

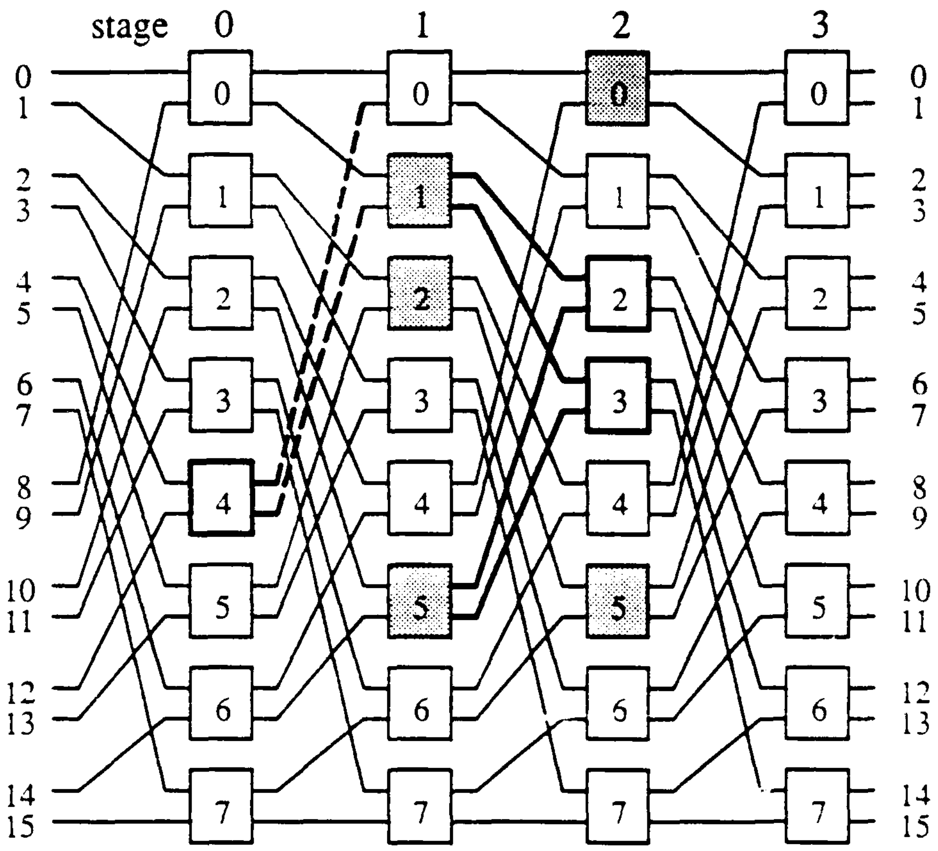


Fig. 5.3. An example fault set F on a 16 × 16 Omega network.

```
                where $n - 1 \geq l > k$, $n - 1 \geq j \geq k$ **do**
                { when $j = k$, bit $y_{-1}$ is undefined. }
                    $mark[\zeta_L] \leftarrow False$
                    $mark[\zeta_{SW}] \leftarrow False$
    { delete those unaccessible processors due to the faulty switching elements }
    for each $\zeta_{SW} = (k, (i_{n-k-2}, i_{n-k-3}, ..., i_0, x_{k-1}, x_{k-2}, ..., x_0)) \in F$ do
        if $mark[\zeta_{SW}] \neq False$ then
            $Z_i \leftarrow Z_i / \{(x_{k-1}, x_{k-2}, ..., x_0, c, ..., c) \mid \#(c) = n - k\}$
            for each
            $\zeta_{SW} = (l, (i_{n-l-2}, i_{n-l-3}, ..., i_0, x_{k-1}, x_{k-2}, ..., x_0, y_{l-k-1}, y_{l-k-2}, ..., y_0)) \in F$,
            $\zeta_L = [l, (i_{n-l-2}, i_{n-l-3}, ..., i_0, x_{k-1}, x_{k-2}, ..., x_0, y_{l-k-1}, y_{l-k-2}, ..., y_0, i_{n-l-1})] \in F$,
            where $n - 1 \geq l > k$ do
                $mark[\zeta_{SW}] \leftarrow False$
                $mark[\zeta_L] \leftarrow False$
    $k \leftarrow k + 1$
```

Note that all the single-pass communication paths starting from processor i construct a binary tree. Algorithm 1 identifies the faulty components in this binary tree stage by stage and deletes all the unaccessible processors due to the destruction of communication by either faulty links or faulty switching elements. Since the failure of a component can destroy the communication from processor i to a subtree rooted at this component, any faulty components in this subtree will not cause the deletion of any new processors. Thus, such faulty components are marked with a value *False* to indicate that no deletion operations are needed to be performed when they are scanned. The total number of faulty components in F is at most $N(\log_2 N - 1) + \frac{N}{2}\log_2 N$. To identify or mark any one of them in the binary tree a $\log_2 N$-bit comparison operation is needed. Thus, the scanning of all the faulty components in F takes a time in $O(N(\log N)^2)$ in the worst case. Also, Algorithm 1 needs at most N deletion operations. Therefore, Algorithm 1 takes a time in $O(N(\log N)^2)$ in the worst case.

Since Z_i is the set of accessible processors of processor i, $0 \le i \le N-1$, We can associate each processor $j \in Z_i$ with a number $[C^*]_{i,j} = 1$ and other processors $k \in \{0, 1, ..., N-1\}/Z_i$ (i.e., the set difference of $\{0, 1, ..., N-1\}$ and Z_i), a number $[C^*]_{i,k} = -1$. Thus, we have defined an array $[C^*]$ where $[C^*]_{i,j}$, $i, j \in \{0,1,...,N\text{-}1\}$, denotes the entry in array $[C^*]$ at the intersection of row i and column j. Obviously, the array $[C^*]$ represents the accessibility of each processor in the first pass through the faulty network, i.e., processor i can communicate with processor j iff $[C^*]_{i,j} = 1$. Note that each processor always can communicate with itself without passing through the network. Even through the diagonal entries $[C^*]_{i,i}$'s in $[C^*]$ may not be all equal to 1, this however will give more convenience for our presentation.

Some processors may lose all the communication paths to or from all the N processors, say, due to faulty switching elements at input stage or output stage. Thus, no possible data packets issued by these dead processors will arrive at other processors or no possible data packets will be received by these dead processors. They can be found by inspecting array $[C^*]$ such that iff they are some i or k such that $[C^*]_{i,j} = -1$ or $[C^*]_{j,k} = -1$, for all $0 \le j \le N-1$, respectively. These processors must be disabled (or conceptually be removed) to avoid blocking other communication and further slowing down the system. Let S_{ur} be the set of surviving processors excluding all the dead processors, referred to as the *surviving system*. Define the following two arrays, *restriction array* and *connection array* to represent the communication capability of the surviving system S_{ur} for the first pass through the faulty network.

DEFINITION 5: Array $[C_R]$, referred to as the restriction array, is defined as $[C_R]_{i,j}$ = $[C^*]_{i,j}$, for all $i, j \in S_{ur}$. Array $[C_S]$, referred to as the connection array, is defined as $[C_S]_{i,i} = 1$ and $[C_S]_{i,j} = [C^*]_{i,j}$, for all $i, j \in S_{ur}$ and $i \neq j$. □

It is clear that $[C_R]$ represents the communication capability of the surviving system S_{ur} without considering the self-communication capability of each surviving processor. on the other hand, $[C_S]$ include the self-communication capability of each surviving processor.

5.3.C. Construction of the Surviving System

The surviving system with respect to a fault set can be constructed if we knew the corresponding maximal fault set. The principle for constructing the surviving system is that: find the maximal fault set corresponding to a fault set and then remove all the components in the maximal fault set and those isolated processors. Thus, the remaining substructure is the surviving system interconnected by the surviving network through which the communication between the surviving processors can be maintained. The isolated processors without any possible incoming or outgoing paths correspond to those dead processors which cannot receive data packets or whose data packets cannot arrive at other processors, respectively. The algorithm to construct the maximal fault set $F_{\max}$ corresponding to a given fault set F can be found in [VaRa89] by Varma and Raghavendra where only faulty switching elements are considered. To fit into our more general fault model where both faulty links and switching elements are considered, we generalize their algorithm as follows.

Algorithm 2:

```
{ Construct the maximal fault set $F_{\max}$ }
procedure Maximal-Fault-Set ($F$ : a fault set)
    $F_{\max} \leftarrow F$    { initially, $F_{\max} = F$ }
```

```
$A_{dd} \leftarrow \phi$   { initially, the set for new additional switching elements is empty }
{ include those links connected to switching elements in $F_{\max}$ }
for each $(i, e) \in F_{\max}$ do
    $F_{\max} \leftarrow F_{\max} \cup \{[i, 2e], [i, 2e+1], [i, e \; div \; 2^{n-2} + 4 \cdot (e \; mod \; 2^{n-2})],$
            $[i, e \; div \; 2^{n-2} + 4 \cdot (e \; mod \; 2^{n-2}) + 2]\}$
{ include those switching elements if the pair of links connected to their
input ports or output ports are in $F_{\max}$ }
for $i = 1$ to $n - 1$ do
    for $e = 0$ to $N/2 - 1$ do
        if $\{[i, 2e], [i, 2e+1]\} \subseteq F_{\max}$ then
            $A_{dd} \leftarrow A_{dd} \cup \{(i, e)\}$
            $F_{\max} \leftarrow F_{\max} \cup \{(i, e)\}$
        if $\{[i, e \; div \; 2^{n-2} + 4 \cdot (e \; mod \; 2^{n-2})], [i, e \; div \; 2^{n-2} + 4 \cdot (e \; mod \; 2^{n-2})$
        $+ 2]\}$
        $\subseteq F_{\max}$ then
            $A_{dd} \leftarrow A_{dd} \cup \{(i-1, e)\}$
            $F_{\max} \leftarrow F_{\max} \cup \{(i-1, e)\}$
{ include buddy pairs of switching elements }
{ forward pass }
for $i = 1$ to $n - 1$ do
    for $w = 0$ to $N/4 - 1$ do
        if $\{(i, w), (i, w + 2^{n-2})\} \subseteq F_{\max}$ then
            $A_{dd} \leftarrow A_{dd} \cup \{(i+1, 2w), (i+1, 2w+1)\}$
            $F_{\max} \leftarrow F_{\max} \cup \{(i+1, 2w), (i+1, 2w+1)\}$
{ reverse pass }
for $i = n - 1$ downto 1 do
    for $w = 0$ to $N/4 - 1$ do
        if $\{(i, 2w), (i, 2w+1)\} \subseteq F_{\max}$ then
            $A_{dd} \leftarrow A_{dd} \cup \{(i-1, w), (i-1, w + 2^{n-2})\}$
            $F_{\max} \leftarrow F_{\max} \cup \{(i-1, w), (i-1, w + 2^{n-2})\}$
{ include links connected to switching elements in $A_{dd}$ }
for each $(i, e) \in A_{dd}$ do
    $F_{\max} \leftarrow F_{\max} \cup \{[i, 2e], [i, 2e+1], [i, e \; div \; 2^{n-2} + 4 \cdot (e \; mod \; 2^{n-2})],$
            $[i, e \; div \; 2^{n-2} + 4 \cdot (e \; mod \; 2^{n-2}) + 2]\}$
```

The maximal fault set $F_{\max}$ is constructed starting from $F_{\max} = F$ by

(1) adding those links connected to switching elements in $F_{\max}$;

(2) adding those switching elements if the pair of links connected to their input ports or output ports is in $F_{\max}$;

(3) scanning the network first from the input side to the output side and then in the reverse order; adding the buddy pairs of all the pairs already in $F_{\max}$, until no more additions are possible,

(4) including those links connected to new additional switching elements in A_{dd} to $F_{\max}$.

It can be proved [VaRa89] that one forward pass and one backward pass are sufficient to include all the buddy pairs deduced from F and thus obtain the corresponding $F_{\max}$. By removing all the components in $F_{\max}$ and those isolated processors, we can obtain the surviving system S_{ur}. As the study in [VaRa89] have shown, for $N \leq 16$ cases, a fault set F is critical with respect to the original N-processor system iff its corresponding maximal fault set $F_{\max}$ contains switching elements from input or output stages of the network. However, to determine the criticality of a fault set F for cases where $N > 16$, no general necessary and sufficient conditions based on the distribution of faulty switching elements have been found yet. The condition that the corresponding maximal fault set $F_{\max}$ contains no switching elements from input or output stages of the network is only necessary for the non-criticality of a fault set F with respect to the original N-processor system. More precisely, we can show that for cases where $N > 16$, even through some processors are removed due to switching elements from input or output stages in $F_{\max}$, the non-criticality of a fault set F with respect to the surviving system S_{ur} still cannot be determined. Nevertheless, we will show in the following sections that to determine the DFA property of the surviving system is not so pessimistic as it looks like. Actually, the information of restriction array $[C_R]$ or connection array $[C_S]$ will be sufficient for determining the DFA property and developing an efficient fault-tolerant routing scheme for the surviving system and the construction of the maximal fault set will not be necessary.

For example, a 32×32 Omega network with a fault set F is shown in Fig. 5.4(a). For easily constructing the surviving system, we show an alternative drawing of the Omega network as in Fig. 5.4(b) which is a butterfly structure. The switching elements {(1,4), (1,12)} and {(1,12), (1,13)} at stage 1 are faulty buddy pairs. As we perform Algorithm 2 and scan the network forward and backward, two switching elements {(0,3), (4,5)} are included in $F_{\max}$ due to that those links connected to them are in $F_{\max}$. And, four other pairs of switching elements are included in $F_{\max}$ due to faulty buddy pairs, i.e., switching elements {(0,6), (0,14), (2,8), (2,9), (3,2), (3,3), (4,6), (4,7)} and those communication links connected to them are included in $F_{\max}$. By removing all the components in the maximal fault set from the network, processors {3, 6, 7, 10, 11, 12, 13, 14, 15, 19, 22, 30, 31} are isolated from the original system. The surviving system {0, 1, 2, 4, 5, 8, 9, 16, 17, 18, 20, 21, 23, 24, 25, 26, 27, 28, 29} is shown in Fig. 5.5.

Based on the construction of the surviving system S_{ur}, an interesting property of the restriction array $[C_R]$ is derived as described in Theorem 1. Theorem 1 states that the data from a number of rows of $[C_R]$ alone will be sufficient to represent $[C_R]$. We will show in the next section that this property can save a lot of computational efforts for a processor to find all the shortest routes to other processors.

THEOREM 1: Let $[C_R]_i$, $i \in S_{ur}$, denote ith row of $[C_R]$ which corresponds to processor i in S_{ur}. If both j and $j + N/2$ are two processors in S_{ur}, $0 \leq j \leq N/2 - 1$, then $[C_R]_j$ and $[C_R]_{j+N/2}$ are identical, i.e., $[C_R]_{j,k} = [C_R]_{j+N/2,k}$, for all $k \in S_{ur}$.

PROOF: Different situations are considered with respect to switching elements and links in $F_{\max}$.

Processor pairs $\{k, k + N/2\}$ or $\{2k, 2k + 1\}$, $0 \leq k \leq N/2 - 1$, which are connected to switching elements in $F_{\max}$ from input or output stages respectively, are removed since they do not belong to S_{ur}. Thus, only switching elements from stage 1 to stage $n-2$ in $F_{\max}$ can affect the communication between processors in S_{ur} and need to be considered. As we have mentioned, any switching element at stage i, $1 \leq i \leq n - 2$, is a common root of two communication trees and 2^{i+1} source processors are connected to 2^{n-i} destination processors through this switching element (for a single pass through the network). If this switching element is in $F_{\max}$, then the outgoing paths from these 2^{i+1} source processors to those 2^{n-i} destination processors will be destroyed. Since $i \geq 1$, at least four source processors are affected by a switching element in $F_{\max}$ at stage i, $1 \leq i \leq n - 2$. That is, any switching element $e = (e_{n-1}, e_{n-2}, ..., e_1)$ at stage i in $F_{\max}$ will destroy outgoing paths to the set of destination processors $\{(e_i, e_{i-1}, ..., e_2, e_1, c, ..., c)\}$, $\#(c) = n - i$, of 2^{i+1} source processors with the common address label equal to $(c, ..., c, e_{n-1}, e_{n-2}, ..., e_{i+1})$, where $2^{i+1} \geq 4$ or $\#(c) = i + 1 \geq 2$. These 2^{i+1} source processors can always be partitioned into four groups such that in each group the first two bits of addresses of all the processors are the same and each processors has the relative address label equal to $(c, ..., c, e_{n-1}, e_{n-2}, ..., e_{i+1})$, $\#(c) = i - 1$. Note that each relative address in each of the four groups has $n - 2$ bits. Each group is a subset of one of

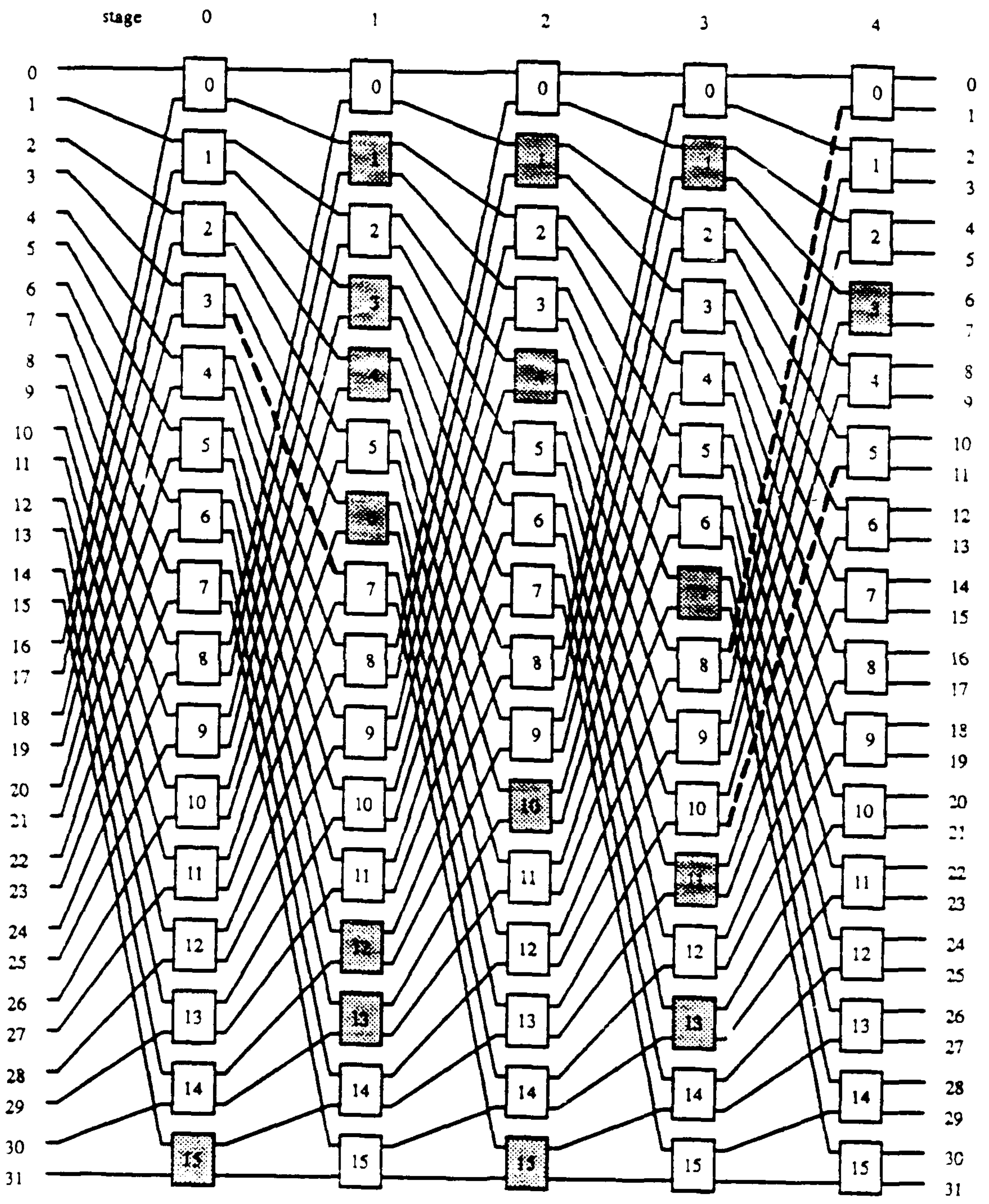


Fig. 5.4(a). A 32 × 32 Omega network with a fault set.

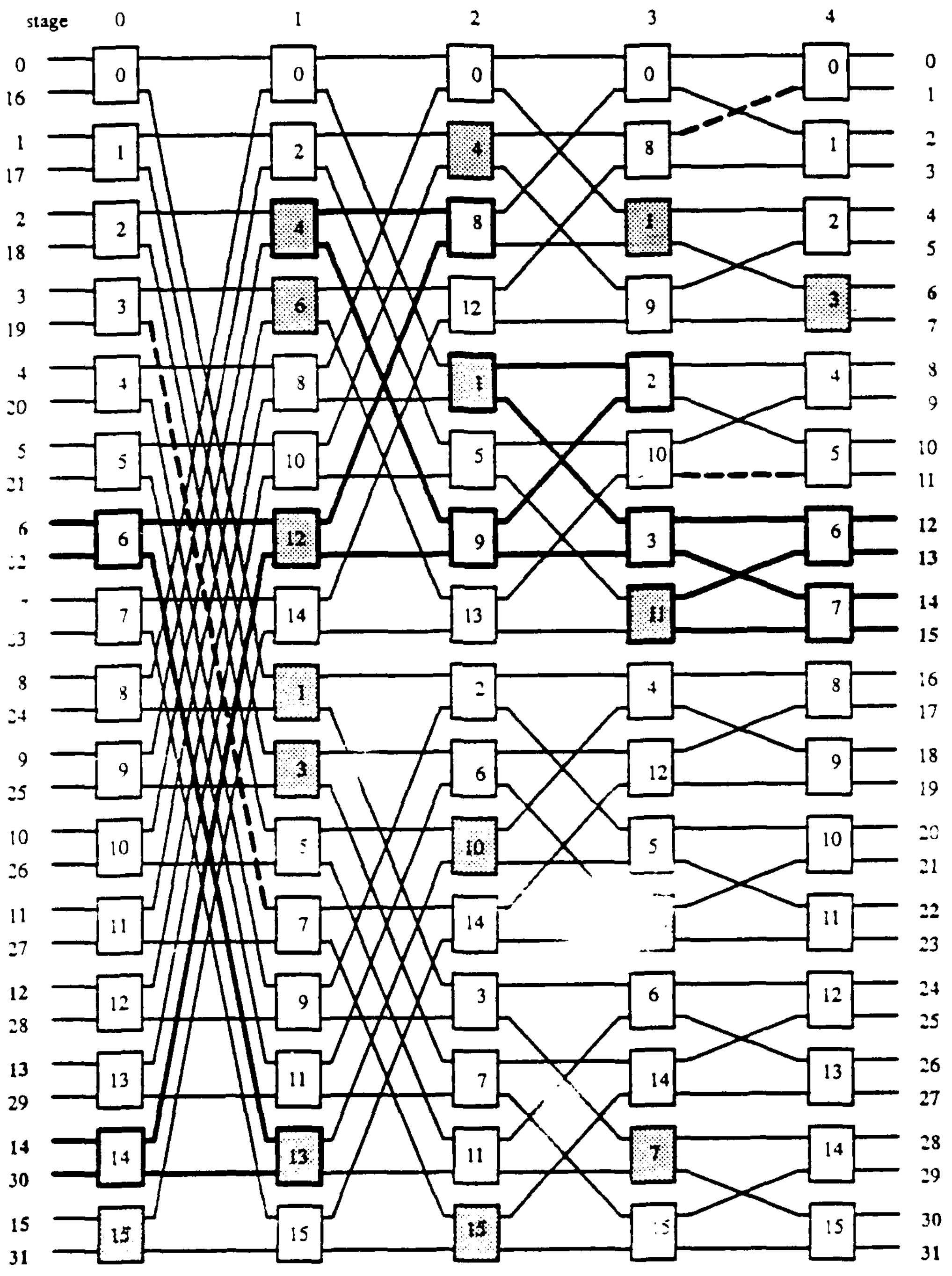


Fig. 5.4(b). An alternative drawing of the Omega network with the same fault set.

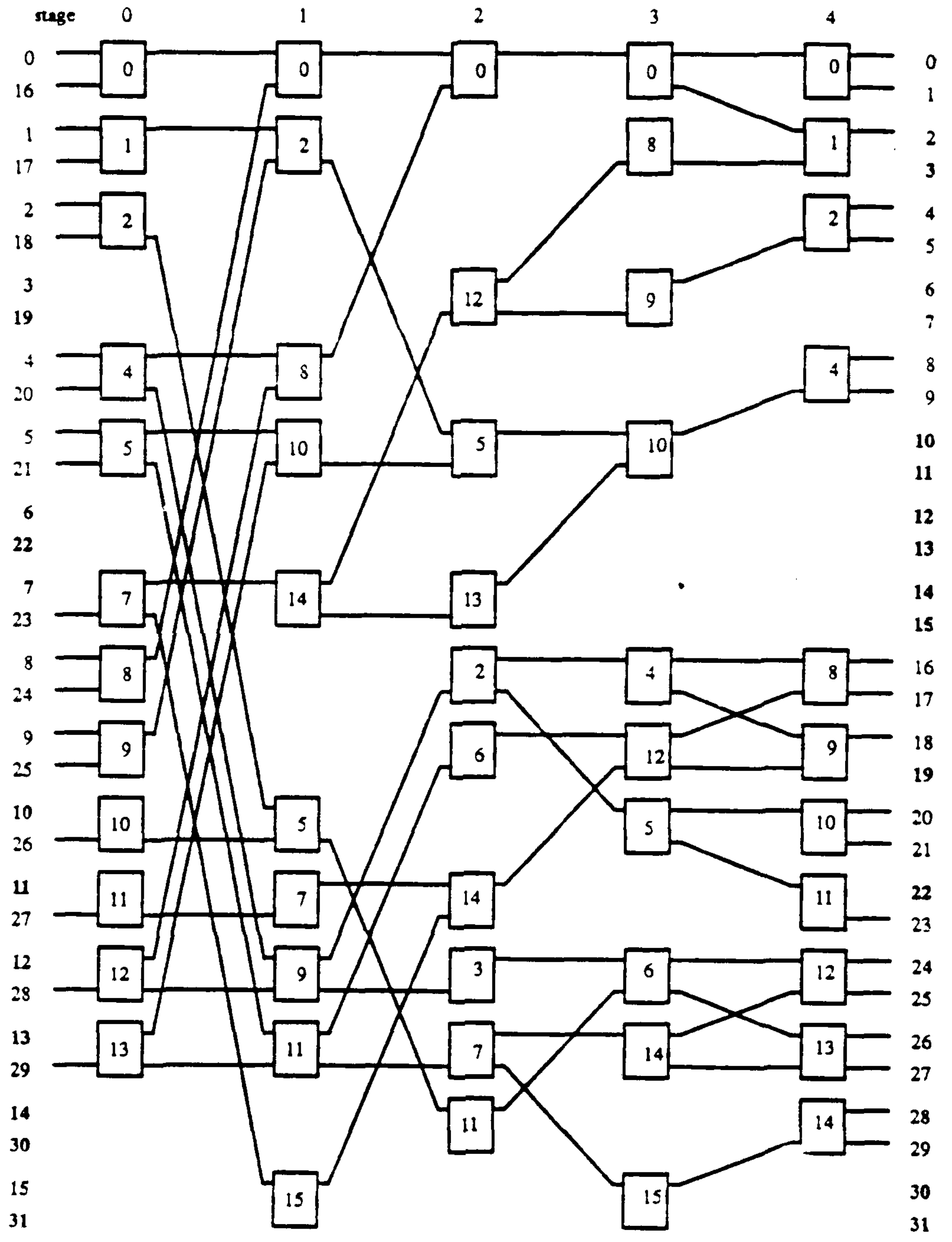


Fig. 5.5. The surviving substructure of the system in Fig. 5.4 after removing all the components in the maximal fault set.

the four subsets of processors $\psi_x = \{x \cdot N/4 + y \mid 0 \le y \le N/4 - 1\}, 0 \le x \le 3$. That is, $\{(0, 0, c, ..., c, e_{n-1}, e_{n-2}, ..., e_{i+1})\} \subset \psi_0 = \{(0, 0, y_{n-3}, y_{n-4}, ..., y_1, y_0)\}$, $\{(0, 1, c, ..., c, e_{n-1}, e_{n-2}, ..., e_{i+1})\} \subset \psi_1 = \{(0, 1, y_{n-3}, y_{n-4}, ..., y_1, y_0)\}$, $\{(1, 0, c, ..., c, e_{n-1}, e_{n-2}, ..., e_{i+1})\} \subset \psi_2 = \{(1, 0, y_{n-3}, y_{n-4}, ..., y_1, y_0)\}$, and $\{(1, 1, c, ..., c, e_{n-1}, e_{n-2}, ..., e_{i+1})\} \subset \psi_3 = \{(1, 1, y_{n-3}, y_{n-4}, ..., y_1, y_0)\}$. Note that not all the 2^{i+1} processors $\{(c, ..., c, e_{n-1}, e_{n-2}, ..., e_{i+1}) \mid \#(c) = i + 1 \ge 2\}$ may exist in S_{ur} since some of them may be removed due to the switching elements in $F_{\max}$ from input or output stages. Therefore, we are ready to make the following conclusions. Any switching element in $F_{\max}$ will destroy the communication from four source processors with the same relative address in each of the four subsets to the same set of destination processors. Of cause, we assume here that two or more of these four source processors exist in S_{ur}, i.e., not all and at most two of them are removed due to the switching elements in $F_{\max}$ from input or output stages, otherwise it becomes a trivial case. Thus, the combined effect of all the switching elements in $F_{\max}$ on these processors in S_{ur} with the same relative address in each of the four subsets is that the resulting communication capability of these processors for a single pass through the network is identical. That is, if $F_{\max}$ contains only switching elements and if $U = \{k + l \cdot N/4 \mid 0 \le l \le 3\} \cap S_{ur} \ne \phi$, $0 \le k \le N/4 - 1$, then for all $x \in U$, $[C_R]_x$'s are identical. It is also true for the statement that if $F_{\max}$ contains only switching elements and both processors j and $j + N/2$ are in S_{ur}, $0 \le j \le N/2 - 1$, then $[C_R]_j$ and $[C_R]_{j+N/2}$ are identical.

Similarly, for any communication link in $F_{\max}$, the communication capability of at least two processors are affected. All the affected processors can be partitioned into two groups. Each group corresponds to the same relative addresses in one of the two subsets of processors $\psi_x = \{x \cdot N/2 + y \mid 0 \le y \le N/2 - 1\}, 0 \le x \le 1$. Therefore, if $F_{\max}$ contains only links and both processors j and $j + N/2 \subseteq S_{ur}$, $0 \leq j \leq N/2 - 1$, then $[C_R]_j$ and $[C_R]_{j+N/2}$ are identical. $\square$

For example, in Table 5.1, the restriction array $[C_R]$ and the connection array $[C_S]$ are shown for the surviving system in Fig. 5.5, where each entry "*" or "□" represent 1 and elsewhere, -1. The results can be checked directly from Fig. 5.5. Moreover, we have $[C_R]_0 = [C_R]_{16}$, $[C_R]_1 = [C_R]_{17}$, $[C_R]_2 = [C_R]_{18}$, and so on. The correctness of Theorem 1 is obvious.

5.3.D. Construction of Shortest-Path Routing Tables

After the restriction array $[C_R]$ or connection array $[C_S]$ of S_{ur} are obtained, it is straightforward to model the multipass routing problem on S_{ur} by using a simple directed multigraph. Obviously, to know whether there exist communication paths between any pair of processors of S_{ur} by going multiple passes through the faulty network is equivalent to determine the reachability between these two vertices on a simple directed multigraph. In order to reconfigure the surviving system in a most efficient way, an appropriate fault-tolerant routing scheme between the surviving processors need to be developed, which is our major concern in this section. We show that a *breadth-first-search* algorithm [PrYe73] can be used to find shortest multi-pass communication paths between any pair of processors in the surviving system. Hereafter, the two terms, communication paths and paths, will be used interchangeably to represent routing paths by one or more passes through the faulty network unless otherwise specified.

Imagine that we have a set V with $\#(S_{ur})$ vertices which are indexed on the set S_{ur}. For each vertex $v_i \in V$, there is a corresponding processor i, $i \in S_{ur}$. A $\#(S_{ur})$-vertex directed multigraph, G, can be constructed as follows: there is an arc $(\overrightarrow{v_i, v_j})$ from v_i to v_j iff $[C_S]_{i,j}$

= 1, where v_i, $v_j \in V$. In other words, the arc $(\overrightarrow{v_i, v_j})$ exists iff processor i can access processor j by the first pass through the faulty network. Of course, the loop $(\overrightarrow{v_i, v_i})$ always exists on each vertex v_i. A *simple* directed multigraph is a graph such that for any two vertices, v_i and v_j, there exists at most one arc from either v_i to v_j or v_j to v_i. Because of the unique-path property of the Omega network, it is very easy to show that the graph G is simple. Also, if we assume that all the single-pass communication paths of the network between any pair of network input and output of the network are equally important, G will be an equally weighted graph with the same weight on all its arcs. Therefore, we have modeled the surviving system S_{ur} by a simple directed multigraph G which is equally weighted.

Define a new array $[C_S^*]$ such that for all i, $j \in S_{ur}$, if $[C_S]_{i,j} = -1$ then $[C_S^*]_{i,j} = 0$; else $[C_S^*]_{i,j} = 1$. A vertex v_j is said to be ***reachable*** from another vertex v_i iff there is a path from v_i to v_j or $[C_S^*]_{i,j}^k \neq 0$, *for some* $k \geq 1$. (here $[C_S^*]^k$ represents the kth power of $[C_S^*]$) The *order* of a path (the number of arcs on the path) connecting v_i to v_j represents the number of passes needed through the faulty network for processor i to access processor j. All the intermediate vertices on the path represent those intermediate processors which need to be traversed. It can be proved [HoSa78] that if vertex v_j is reachable from vertex v_i on G, then $[C_S^*]_{i,j}^k \neq 0$ in at least one k where $1 \leq k \leq \#(S_{ur}) - 1$. The graph G is said to be *strongly connected* if for each pair of vertices v_i and v_j, there exists at least one path from v_i to v_j and one path from v_j to v_i. Thus, G is strongly connected iff the surviving system S_{ur} has the DFA property. Generally speaking, to determine the DFA property of S_{ur} we need to traverse the faulty network at most $\#(S_{ur}) - 1$ passes and check each entry of each array $[C_S^*]^k$, i.e., determine the reachability between any pair of processors. A similar method is used in [AgLe85] to understand the DFA property of multiprocessor interconnected by a

Table 5.1. The restriction array $[C_R]$ and the connection array $[C_S]$ for the surviving system in Fig. 5.5.

	0	1	2	4	5	8	9	16	17	18	20	21	23	24	25	26	27	28	29
0	*	*	*																
1		■				*	*												
2			■											*	*	*	*		
4	*	*	*	■				*	*	*	*	*	*	*	*	*	*		
5					■	*	*	*	*	*	*			*	*	*	*	*	*
8	*	*	*			■													
9						*	*												
16	*	*	*					■											
17						*	*		■										
18										■				*	*	*	*		
20	*	*	*					*	*	*	*	*	*	*	*	*	*		
21						*	*	*	*	*	*	■		*	*	*	*	*	*
23			*	*	*	*	*	*	*	*			■						
24	*	*	*											■					
25						*	*								■				
26														*	*	*	*		
27								*	*	*									
28	*	*	*					*	*	*	*	*	*	*	*	*	*	■	
29						*	*	*	*	*	*			*	*	*	*	*	*

$[C_R] = \{ * \}$ $\qquad$ $[C_S] = \{ * \} + \{ \blacksquare \}$

faulty network. However, the computational complexity is prohibitively high as the size of system increases.

As long as the shortest communication paths are used for communication between processors in the surviving system, an appropriate shortest-path routing table for a processor to access other processors must be supplied. In order to access a destination processor, the corresponding entry of the shortest-path routing table for a source processor must include the following information:

(1) the proper intermediate processors through which its data packets will be routed in the first pass through the faulty network,

(2) the minimum number of passes through the faulty network to arrive at the destination processor.

Therefore, the shortest-path fault-tolerant routing scheme on the surviving system is described as follows. Whenever a data packet arrives at an intermediate processor after a pass through the network, the control portion of this data packet contains

(1) the source and destination addresses (which will not be changed during communication),

(2) the number of passes left to reach the destination processor,

(3) the address of next intermediate processor (there may exist many possible ones) for the next pass through the network; this address is appended after the entry corresponding to the destination processor of the shortest-path routing table of the current intermediate processor has been referred to.

The address of next intermediate processor is used as the temporary routing tag for the next pass through the network if it is not equal to the destination address. The number of passes left to reach the destination processor will be subtracted by one after the next pass through the network and compared with that in the shortest-path routing table of next intermediate processor for advanced fault-tolerant control. Thus, the information of the intermediate processors for the first pass given in each entry of a shortest-path routing table will be sufficient to support this kind of fault-tolerant routing scheme. It is clear that the bit-oriented routing scheme of the original system (i.e., the Omega network) has been preserved in the surviving system.

By Algorithm 1 and some data manipulation, the connection array $[C_S]$ is obtained. The data of $[C_S]$ is then broadcast from the reconfiguration unit to each processor in S_{ur} where a breadth-first-search algorithm is performed to find the shortest-path routing table for each processor itself. Thus, the advantage is that an identical breadth-first-search procedure using an identical set of input data is executed in parallel in each processor of S_{ur}. To implement the breadth-first-search algorithm, we need a type of data structure *queue* that allows two operations *enqueue* and *dequeue*. This type represents a list of elements that are to be handled in a first-come-first-serve manner. The function of $first(Q)$ denotes the element at the front of the queue Q. According to the shortest-path routing scheme, for a processor i following the shortest paths to access the destination processor j, the information of proper intermediate processors of the first pass through the faulty network will be sufficient. Thus, to access processor j, a set *intermediate*$[j]$ is used to include all the possible intermediate processors of the first pass and *passes*$[j]$ is used to indicate the minimum number of passes required through the faulty network. Initially, *intermediate*$[j]$ is an empty set and *passes*$[j] = 1$, for all $j \in S_{ur}$. An index set I is used to contain all the intermediate processors which processor i can reach in the first pass through the faulty network. Eventually, the shortest-path routing table of processor i is given by ith row of an array $[A_S]$, i.e., $[A_S]_{i,j}$ =

($intermediate[j]$, $passes[j]$)$_i$ = $(\alpha_{i,j}, \beta_{i,j})$ gives the routing information to access processor j. Thus, if $intermediate[j] \neq \phi$ in $[A_S]_{i,j}$, then this means that processor i can access processor j by routing its data packets through any one of the intermediate processor in $intermediate[j]$ in the first pass and that its data packets will arrive at processor j in the minimum $passes[j]$ passes through the faulty network. However, if $intermediate[j] = \phi$ for at least one j, then this means that processor i cannot access all the processors in S_{ur}. Algorithm 3 gives the procedure to construct the shortest-path routing table for a processor i in the surviving system S_{ur}.

Algorithm 3:

```
{ Construct the shortest-path routing table for each processor in $S_{ur}$ }
procedure Breadth-First-Search ($i \in S_{ur}$: processor; $[C_S]$: array)
  $Q \leftarrow \phi$   { empty queue }
  $I \leftarrow \phi$   { empty set of intermediate processors }
  for each $j \in S_{ur}$ such that $[C_S]_{i,j} \neq -1$ do
    $I \leftarrow I \cup \{j\}$
    $intermediate[j] \leftarrow intermediate[j] \cup \{j\}$
    enqueue $j$ into $Q$
  { loop to find shortest paths }
  if $I \neq S_{ur}$ then
    while $Q \neq \phi$ do
      $j \leftarrow first(Q)$
      dequeue $j$ from $Q$
      for each $k$ such that $[C_S]_{j,k} \neq -1$ do
        if ($passes[k] = 1$ or $passes[k] - 1 = passes[j]$) and $k \in S_{ur}/\{I\}$
        then
          $intermediate[k] \leftarrow intermediate[j] \cup intermediate[k]$
          $passes[k] \leftarrow passes[j] + 1$
        enqueue $k$ into $Q$
  { the shortest-path routing table }
  for each $j \in S_{ur}$ do
```

$$[A_S]_{i,j} \leftarrow (intermediate[j], passes[j])_i$$

The correctness of Algorithm 3 is discussed as follows.

Algorithm 3 is essentially a procedure applying a breadth-first-search on the multigraph G as mentioned above. Hence, to obtain the minimal number of passes through the faulty network for processor i to access other processors, we simplely traverse the multigraph G starting from v_i (processor i) using a breadth-first-search. That is to say, we start from v_i and visit all the sons of v_i, then visit all the grandsons of v_i, and so on. The visiting continues until all the visitable vertices have been visited. It can be shown that for v_j to be visited, the paths from v_i via its parents are shortest. Thus, the first step of shortest paths from v_i to v_j is the same as the first step of shortest paths from v_i to parents of v_j. Therefore, the shortest paths from v_i (processors i) to all the visitable vertices (processors) are traversed. Moreover, the breadth-first-search searches all the possible descendent vertices (processors) from vertex i. If at a level of search no new son vertices are visited, it means that a subset of previously visited vertices will be visited again. Since all the son vertices of these previously visited vertices have been extensively searched, no possible paths exist from these visited vertices to other new vertices. That is to say, the search is terminated if at a level of search no new son vertices are visited (all the searching branches become loops). The above argument gives the proof of the fact: there are no paths from vertex i to vertex j iff there are no shortest paths from vertex i to vertex j by Algorithm 3. Assume that the total number of arcs traversed is E when Algorithm 3 is implemented starting from a vertex on G. It is easy to show that the time complexity of Algorithm 3 (which is a breadth-first-Search algorithm) is $O(E + N) = O(max(E,N))$. We will prove that $O(max(E,N)) = O(N^2)$ in Section 5.4 where the overall time complexity of our fault-tolerant reconfiguration scheme is discussed.

According to Theorem 1, a lot of computational efforts can be saved by using the restriction array $[C_R]$ instead of the connection array $[C_S]$. It is clear that in the worst case where the size of S_{ur} is N, the first half of rows of $[C_R]$, i.e., $\{[C_R]_i \mid 0 \le i \le N/2 - 1\}$, will provide enough information to find all the shortest-path routing tables. Moreover, at most $N/2$ processors will be sufficient for computing all the N shortest-path routing tables.

For example, in Table 5.2, the shortest-path routing tables associated with the faulty Omega network in Fig. 5.5 are shown. For simplicity, only the parameter $\beta_{i,j}$ (the minimal number of passes) of an entry $[A_S]_{i,j}$ is shown in Table 5.2.

5.3.E. Reconfiguration of The Surviving System

By Algorithm 3, each processor in the surviving system obtains its shortest-path routing table. In some situations, some processors may only be able to access a part of processors in the surviving system. For example, in Fig. 5.5, processor 1 cannot access processors {4, 5, 20, 21, 23, 28, 29}. Obviously, such situations are due to the criticality of the fault set with respect to the surviving system. An implicit deadlock has arisen under such circumstances and shortest-path routing tables become traps if they are not used cautiously. Let us consider the following case on the surviving system in Fig. 5.5. Processor 5 can dynamically fully access all the processors in S_{ur}. It is clear that if data packets from processor 5 are sent to processor 1, no possible acknowledge signals will be received by processor 5. That is to say, there will be a deadlock between processor 1 and processor 5. Such a situation must be avoided. Therefore, from the viewpoint of a reliable reconfigurable environment, a deadlock-free reconfiguration algorithm must be employed so that only the bidirectional communication is maintained on the surviving system. Central to the design of such a deadlock-free

Table 5.2. The simplified shortest-path routing tables of the surviving system in Fig. 5.5.

	0	1	2	4	5	8	9	16	17	18	20	21	23	24	25	26	27	28	29
0	*	*	*			+	+	●	●	●				+	+	+	+		
1	+	*	+			*	*	■	■	■				●	●	●	●		
2	+	+	*			+	+	+	+	+				*	*	*	*		
4	*	*	*	*	+	+	+	*	*	*	*	*	*	*	*	*	*	+	+
5	+	+	+	●	*	*	*	*	*	*	*	+	+	*	*	*	*	*	*
8	*	*	*			*	+	●	●	●				+	+	+	+		
9	+	+	+			*	*	■	■	■				●	●	●	●		
16	*	*	*			+	+	*	●	●				+	+	+	+		
17	+	+	+			*	*	■	*	■				●	●	●	●		
18	+	+	+			+	+	+	+	*				*	*	*	*		
20	*	*	*	+	+	+	+	*	*	*	*	*	*	*	*	*	*	+	+
21	+	+	+	●	●	*	*	*	*	*	*	*	+	*	*	*	*	*	*
23	+	+	*	*	*	*	*	*	*	*	+	+	*	+	+	+	+	+	+
24	*	*	*			+	+	●	●	●				*	+	+	+		
25	+	+	+			*	*	■	■	■				●	*	●	●		
26	+	+	+			+	+	+	+	+				*	*	*	*		
27	+	+	+			+	+	*	*	*				+	+	+	*		
28	*	*	*	+	+	+	+	*	*	*	*	*	*	*	*	*	*	*	+
29	+	+	+	●	●	*	*	*	*	*	*	+	+	*	*	*	*	*	*

* : the first pass
+ : the second pass
● : the third pass
■ : the fourth pass

procedure is the sacrifice of some usable links or switching elements and the partitioning of the surviving system into a number of subsystems such that each of which possesses the DFA property (i.e., the fault set is noncritical with respect to each subsystem). In order to utilize the surviving system in a most efficient way, a subsystem should be a maximal disjoint set which only includes all the possible processors with bidirectional communication capability among them. Such a deadlock-free reconfiguration algorithm is our goal in this section.

The reconfiguration monitor is notified with the criticality of the fault set whenever a surviving processor finds itself cannot access some processors in the surviving system, i.e., for some $i, j \in S_{ur}$, $\alpha_{i,j} = \phi$. All the shortest-path routing tables are collected by the reconfiguration monitor where array $[A_S]$ will be inspected. According to the following definition, the *reachability* array $[H_S]$ corresponding to S_{ur} is obtained.

DEFINITION 6: $[H_S]$ is a $\#(S_{ur}) \times \#(S_{ur})$ boolean array associated with $[A_S]$ such that for $i, j \in S_{ur}$, entry $[H_S]_{i,j}$ is a boolean constant which is either *True* or *False*. $[H_S]_{i,j}$ is defined as follows: $[H_S]_{i,j} = False$ if $\alpha_{i,j} = \phi$; otherwise $[H_S]_{i,j} = True$. □

Each $[H_S]_{i,j} = True$ indicates that there exists at least one path from processor i to processor j. To avoid the problem of deadlock, the use of unidirectional communication paths between two processors must be prohibited. For example, if $[H_S]_{i,j} = True$ but $[H_S]_{j,i} = False$, then paths from processor i to processor j should not be used again. A new array $[H_S^*]$ is employed to monitor the status of bidirectional communication between any two processors in S_{ur}. Array $[H_S^*]$ is defined as follows.

DEFINITION 7: $[H_S^*]$ is a $\#(S_{ur}) \times \#(S_{ur})$ boolean array associated with array $[H_S]$ such that for $i, j \in S_{ur}$, entry $[H_S^*]_{i,j} = [H_S^*]_{j,i} = [H_S]_{i,j} \wedge [H_S]_{j,i}$, where $\wedge$ is a boolean *AND* operation. □

Thus, array $[H_S^*]$ is a symmetric array with diagonal elements equal to *True*. For any $i, j \in S_{ur}$, $[H_S^*]_{i,j} = True$ iff there exist bidirectional communication paths between processor i and processor j. It is obvious that array $[H_S^*]$ consists of a number of "*True*" blocks in which all the entries are equal to *True*. The method to construct all the possible maximal disjoint sets in S_{ur} is based on the following argument: a processor i belongs to a disjoint set C_j which must be maximal iff $[H_S^*]_{i,k} = True$, for all $k \in C_j$, and $[H_S^*]_{i,l} = False$, for all $l \in S_{ur}/C_j$. Thus, the subsystem C_j is the maximal disjoint set which contains processor i and possesses the DFA property, i.e., the fault set is noncritical with respect to C_j. This argument is formally described by the following theorem.

THEOREM 2: Assume that there exists an $i \in S_{ur}$ and $T \subseteq S_{ur}$ such that $[H_S^*]_{i,k} = True$, for all $k \in T$, and $[H_S^*]_{i,l} = False$, for all $l \in S_{ur}/T$. Then, for any $x, y \in T$ and $z \in S_{ur}/T$, $[H_S^*]_{x,y} = True$ and $[H_S^*]_{x,z} = False$. Moreover, T is the maximal disjoint set which contains processor i and possesses the DFA property.

PROOF: Since $[H_S^*]$ is a symmetric array with diagonal elements equal to *True*, $i \in T$ is always true. If $[H_S^*]_{i,k} = True$, for all $k \in T$, and $[H_S^*]_{i,l} = False$, for all $l \in S_{ur}/T$, then $[H_S^*]_{k,i} = True$ and $[H_S^*]_{l,i} = False$. Thus, for any $x,y \in T$, there exist communication paths both from x to y and from y to x. That is, by Algorithm 2, shortest paths either from x to y or y to x have been found. Therefore, $[H_S^*]_{x,y} = True$, for all $x,y \in T$. However, if there exists a $z \in S_{ur}/T$ such that $[H_S^*]_{x,z} = True$, for some $x \in T/\{i\}$, this means that there exist communication paths from z via x to i, i.e., $[H_S^*]_{i,z} = True$ which contradicts our assumption. Therefore, $[H_S^*]_{x,z} = False$, for all $z \in S_{ur}/T$ and all $x \in T$. Moreover, elements of T

are all the possible processors possessing bidirectional communication capability with processor i. This is based on the fact that array $[H_S^*]$ is a modified array derived from the reachability array $[H_S]$. Therefore, T is the maximal disjoint set which contains processor i and possesses the DFA property. □

By Theorem 2, the maximal disjoint set C_j to which processor i belongs is composed of those processors k's such that $[H_S^*]_{i,k} = True$. Thus, the shortest-path routing table of processor i can become a deadlock-free one by modifying $[A_S]_i$ as follows.

```
{ modify shortest-path routing tables to deadlock-free ones }
procedure Update ($i \in S_{ur}$: processor; $[H_S^*]$: boolean array)
    for each $j \in S_{ur}$ such that $[H_S^*]_{i,j} = False$ do
        $[A_S]_{i,j} \leftarrow (\phi, 0)_i$
```

For example, the updated routing tables $[A_S]$ for the surviving system in Fig. 5.5 is shown in Table 5.3. Those unidirectional communication paths from, say, processor 20 to processors {0, 1, 2, 8, 9, 16, 17, 18, 24, 25, 26, 27} are discarded.

However, for the reconfiguration monitor to obtain the global information of utilization of the surviving system, a better way to implement the deadlock-free reconfiguration algorithm is described as follows. The work of reconfiguring the surviving system into smaller subsystems is essentially to group processors in the surviving system into maximal disjoint sets such that in each set the DFA property (i.e., the bidirectional communication capability) is preserved. Initially, the $\#(S_{ur})$ processors are in $\#(S_{ur})$ different sets. A *canonical* object, named $set[i]$, is chosen to serve as the label for the set of processor i, $i \in S_{ur}$. Since there is no preference for the choice of labels as long as they are canonical, it is natural that the address of a processor is sufficient to serve as the label, i.e., $set[i] = i$ initially. A reference processor *nonzero* $[i]$ for processor i contains the address of an arbitrary processor j such that $[H_S^*]_{i,j} = 1$ and $i \neq j$. The reconfiguration starts randomly from any processor in S_{ur} by searching and changing the label of another processor to its label iff there exist bidirectional communication paths between them. A processor k is included in a disjoint set iff the label of its reference processor has not been changed, i.e., $set[nonzero[k]] = nonzero[k]$. Eventually, all the labels of processors in a maximal disjoint set will be the same and are equal to the label of some arbitrary processor in the maximal disjoint set. The following is the deadlock-free reconfiguration algorithm.

Algorithm 4:

```
{ Reconfiguration of The Surviving System }
**procedure** Deadlock-Free Reconfiguration ($S_{ur}$: the surviving system; $[H_S^*]$: boolean array)
  $Q \leftarrow \phi$   { empty queue }
  $N \leftarrow 0$
  **for** each $i \in S_{ur}$ **do**
    *enqueue* $i$ into $Q$
  { find maximal disjoint sets $C_j$'s }
  **while** $N \neq \#(S_{ur})$ **or** $Q \neq \phi$ **do**
    $j \leftarrow first(Q)$
    $k \leftarrow nonzero[j]$
    **if** $set[k] = k$ **then**
      **for** each $l$ such that $[H_S^*]_{j,l} = True$ **do**
        $C_j \leftarrow C_j \cup \{l\}$
        $set[l] \leftarrow j$
      $N \leftarrow N + \#(C_j)$
    *dequeue* $j$ from $Q$
```

Algorithm 4 scans each processor in S_{ur} (they may be selected randomly) by searching all the possibly processors for a maximal disjoint set until all the $\#(S_{ur})$ processors have been classified into different maximal disjoint sets. To distinguish maximal disjoint sets which

Table 5.3. The modified shortest-path routing tables $[A_s]$ of the surviving system in Fig. 5.5.

	0	1	2	4	5	8	9	16	17	18	20	21	23	24	25	26	27	28	29
0	*	*	*			+	+	●	●	●				+	+	+	+		
1	+	*	+			*	*	■	■	■				●	●	●	●		
2	+	+	*			+	+	+	+	+				*	*	*	*		
4				*	+						*	*	*					+	+
5				●	*						*	+	+					*	*
8	*	*	*			*	+	●	●	●				+	+	+	+		
9	+	+	+			*	*	■	■	■				●	●	●	●		
16	*	*	*			+	+	*	●	●				+	+	+	+		
17	+	+	+			*	*	■	*	■				●	●	●	●		
18	+	+	+			+	+	+	+	*				*	*	*	*		
20				+	+						*	*	*					+	+
21				●	●						*	*	+					*	*
23				*	*						+	+	*					+	+
24	*	*	*			+	+	●	●	●				*	+	+	+		
25	+	+	+			*	*	■	■	■				●	*	●	●		
26	+	+	+			+	+	+	+	+				*	*	*	*		
27	+	+	+			+	+	*	*	*				+	+	+	*		
28				+	+						*	*	*					*	+
29				●	●						*	+	+					*	*

*	: the first pass	●	: the third pass
+	: the second pass	■	: the fourth pass

have been previously found from the maximal disjoint set which is being currently constructed, we need only check whether or not the *nonzero* parameter of a processor currently searched has been changed (based on the fact of Theorem 2). After the implement of Algorithm 4, the information of a maximal disjoint set is sent back from the reconfiguration monitor to each processor of the maximal disjoint set. Then, the shortest-path routing table of a processor is updated to a deadlock-free one according to the maximal disjoint set to which the processor belongs.

```
{ updates shortest-path routing tables to deadlock-free ones }
procedure Update ($C_i$: maximal disjoint set; $j \in C_i$: processor)
    for each $k \in S_{ur}/C_i$ do
        $[A_S]_{j,k} \leftarrow (\phi, 0)_j$
```

After all the shortest-path routing tables (i.e., array $[A_S]$) has been modified, a number of usable links and switching elements were implicitly sacrificed. These components construct those unidirectional communication paths which were discarded as array $[H_S^*]$ was derived from $[H_S]$. Also, a number of subsystems are formed from the original surviving system. Each maximal disjoint set corresponds to such a subsystem.

For example, by Algorithm 4, the surviving system in Fig. 5.5 is partitioned into two subsystems with DFA property. These two subsystems are {0, 1, 2, 8, 9, 16, 17, 18, 24, 25, 26, 27} and {4, 5, 20, 21, 23, 28, 29}, respectively. The partitioning results are shown in Fig. 5.6. Implicitly, those usable switching elements {(1,8), (1,10), (1,15), (2,13), (3,4), (3,8), (3,14)} and some usable links connected to them are sacrificed.

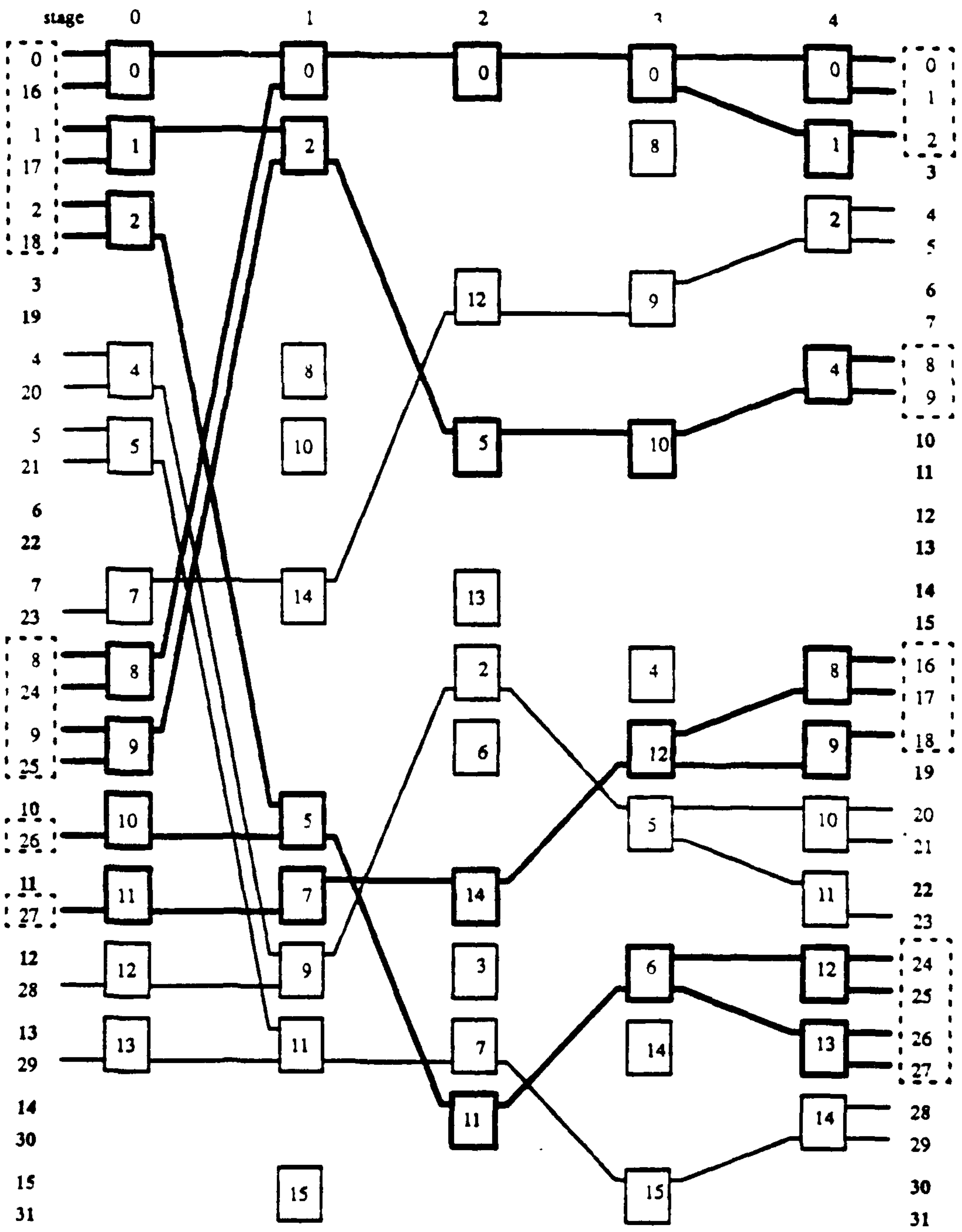


Fig. 5.6. Two subsystems are formed from the surviving system in Fig. 5.5.

5.4. COMPLEXITY OF THE FAULT-TOLERANT RECONFIGURATION SCHEME

The time complexity of the proposed fault-tolerant reconfiguration scheme is analyzed as follows. Assume that the time overhead spent on data communication between reconfiguration monitor and processors is negligible.

(1) Algorithm 1 (the Accessibility) takes a time in $O(N(\log N)^2)$.

(2) The manipulation of a variety of arrays: $[C^*]$, $[C_R]$, $[C_S]$, $[H_S]$ and $[H_S^*]$, takes a time in $O(N^2)$.

(3) Algorithm 3 (the breadth-first-search) takes a time in $O(N^2)$. (We will explain this later.)

(4) Algorithm 4 (the Deadlock-Free Reconfiguration) takes a time in $O(N)$.

(5) Updating the shortest-path routing table for a processor takes a time in $O(N)$.

Therefore, the time complexity is dominated by the time spent on the manipulation of a variety of arrays and Algorithm 3 which are in $O(N^2)$. The following gives the detailed proof of the time complexity of Algorithm 3 which is a breadth-first-search on a directed multigraph.

Algorithm 3 searches all the accessible processors for a surviving processor by a minimal number of passes through the faulty network. It is obvious that the complexity is equal to the number of accessible processors plus the total number of single-pass communication paths traversed (i.e., the total number of arcs traversed on G) to search these accessible processors. The number of accessible processors for a surviving processor is at most N which is straightforward. However, to calculate the number of required single-pass communication through the faulty network needs some sophisticated efforts. The difficulty arises from that the variation of the number of single-pass communication paths traversed is closely related to the distribution of faulty components and the closed form of their relationship is hard to get. Thus, an approximate method might be used to find an upper bound. Assume that a processor can access all the N processors by at least $k+1$ passes through the faulty network, $1 \le k \le N - 1$. That is, by Algorithm 2, $(N - x_1)$ processors are searched by the first pass, $(x_1 - x_2)$ processors are searched by the second pass, $\cdots$, $(x_{k-1} - x_k)$ processors are searched by the kth pass, and x_k processors are searched by the $(k+1)$th pass, where $N > x_1 > x_2 > x_3 > \cdots > x_{k-1} > x_k > 0$ and all x_i's are integers. For the first pass, the number of single-pass paths traversed is equal to $(N - x_1)$. For the second pass, not all the $(N - x_1)$ intermediate processors which have been searched in the first pass can access all the $(x_1 - x_2)$ processors. In general, part of these $(x_1 - x_2)$ processors are searched by routing through part of those $(N - x_1)$ intermediate processors and another part, by some part of others. Because of the symmetric structure of an Omega network, we may think that those x_1 processors which cannot be searched by the first pass are uniformly distributed over the $(N - x_1)$ intermediate processors. Hence, a reasonable estimate of the portion of the $(N - x_1)$ processors through which the $(x_1 - x_2)$ processors can be searched by the second pass is $(N - x_1)/N$. Therefore, an estimated upper bound of the number of single-pass paths traversed by the second pass is

$$\left(\frac{N - x_1}{N}\right)\cdot(N - x_1)\cdot(x_1 - x_2).$$

Similarly, an estimated upper bound of the number of single-pass paths traversed by the ith pass, $1 \le i \le k$ is

$$\left(\frac{x_{i-2} - x_{i-1}}{x_{i-2}}\right)\cdot(x_{i-2} - x_{i-1})\cdot(x_{i-1} - x_i)$$

and the estimated upper bound of the number of single-pass paths traversed by the $(k+1)$th pass is

$$\left(\frac{x_{k-1} - x_k}{x_{k-1}}\right)\cdot(x_{k-1} - x_k)\cdot(x_k).$$

Now the complexity analysis of Algorithm 3 can be modeled as the following problem. For an arbitrary distribution of a fault set, there exists a number k, $1 \le k \le N - 1$, such that by Algorithm 3 a surviving processor can search all the accessible processors (either all or a part of the N processors) by $k+1$ passes through the faulty network. Let the number of accessible processors be $M \le N$ and E denote the total number of single-pass communication paths traversed by Algorithm 3. It can be shown from the above argument that E is bounded by the following equations:

$$\begin{aligned} E \le\ & M - x_1 + \left(\frac{M - x_1}{M}\right)\cdot(M - x_1)\cdot(x_1 - x_2) + \\ & \left(\frac{x_1 - x_2}{x_1}\right)\cdot(x_1 - x_2)\cdot(x_2 - x_3) \\ & + \cdots + \left(\frac{x_{k-1} - x_k}{x_{k-1}}\right)\cdot(x_{k-1} - x_k)\cdot(x_k) \\ & = f(x_1, x_2, \ldots, x_k) \\ & = f(\mathbf{x}), \end{aligned}$$

where

$$M > x_1 > x_2 > x_3 > \cdots > x_{k-1} > x_k > 0,$$

$$x_i, \text{ for all } 1 \le i \le k, \text{ are integers.}$$

Thus, the analysis of complexity of Algorithm 3 has become the constrainted optimization problem where we want to find the maximum of a set of nonlinear functions with inequality constraints. In general, the nonlinear optimization technique [WiCh78] may be employed to find the maximal value among all the possible functions $f(\mathbf{x})$ with inequality constraints which will be the upper bound of E. That is, we have an optimization problem whose solution gives the upper bound of E: for each $1 \le k \le N-1$, find

$$\max_{\mathbf{x}} f(\mathbf{x})$$

such that

$$g_i(\mathbf{x}) < 0, \quad i = 1,2, \ldots, k, k+1$$

where

$$g_i(\mathbf{x}) = \begin{cases} x_1 - N, & i = 1 \\ x_i - x_{i-1}, & 2 \le i \le k. \\ -x_k, & i = k+1 \end{cases}$$

However, a simpler way to obtain the upper bound is discussed as follows.

It is obvious that

$$h(\mathbf{x}) = M - x_1 + (M - x_1)\cdot(x_1 - x_2) + (x_1 - x_2)\cdot(x_2 - x_3) + \cdots + (x_{k-1} - x_k)\cdot(x_k)$$
$$\ge f(\mathbf{x}).$$

To find the maximum among functions $h(\mathbf{x})$, a geometrical method is used here. See Fig. 5.7. It is easy to show that the total area of those shadowed rectangles is equal to $h(\mathbf{x})$. For any k and any arrangement of the values of x_i's, this total area is always less or equal to $M^2/2$. That is,

$$\frac{M^2}{2} \ge \max_{\mathbf{x}} h(\mathbf{x}) \ge \max_{\mathbf{x}} f(\mathbf{x}).$$

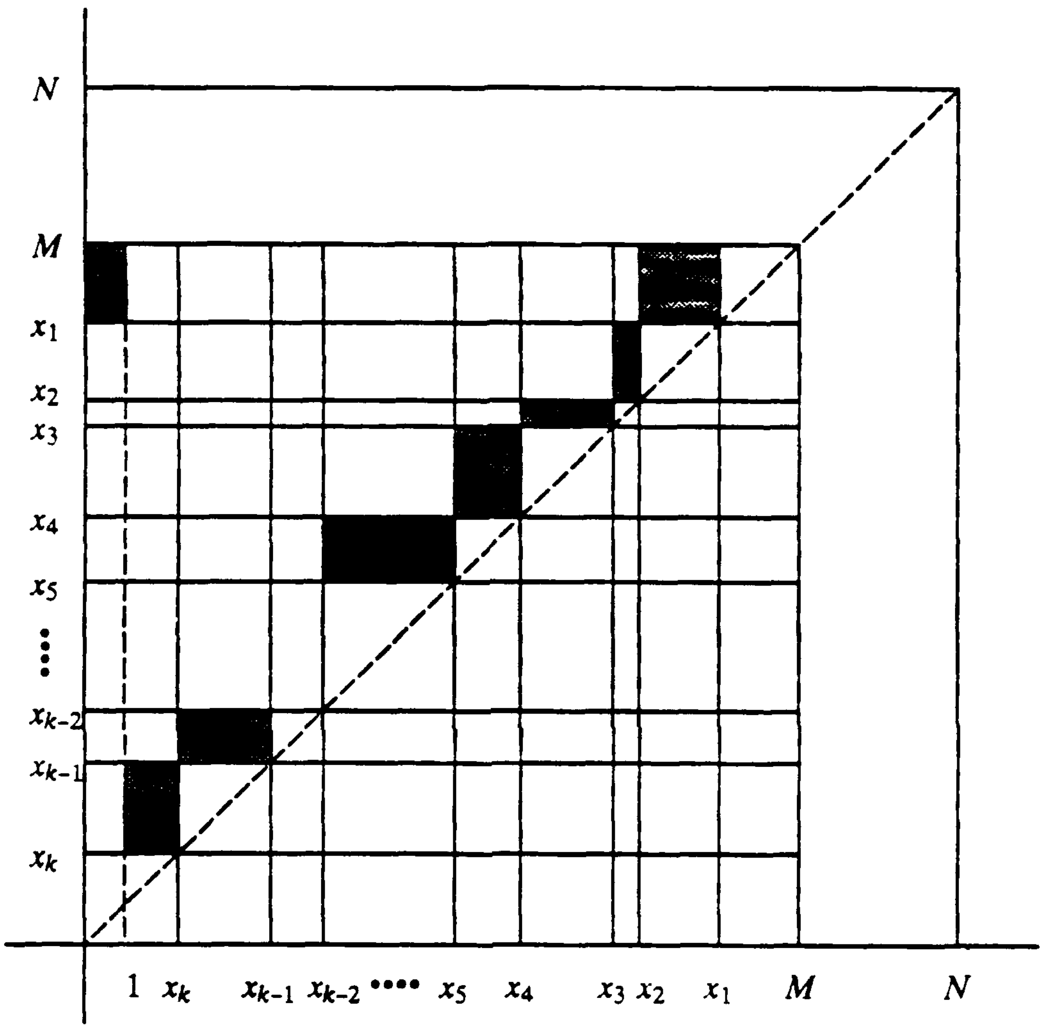


Fig. 5.7. The diagram used to explain the complexity of Algorithm 3.

And the maximal value of M is N which means that the set of accessible processors of a processor by multiple passes through the faulty network is all the N processors. Therefore, the upper bound of E is in $O(N^2)$. Obviously, the above argument gives the proof that Algorithm 3 takes a time in $O(N^2)$.

5.5. SUMMARY

What we have presented in this chapter is a flexible and real-time reconfiguration scheme for a multiprocessor system with a faulty network. Even though our fault-tolerant reconfiguration scheme is developed for an N-processor system interconnected by a $\log_2 N$-stage Omega network, it can be easily extended to a system interconnected by other networks which are topologically equivalent to the Omega network [Agr83] and are constructed by switching elements with different size. Moreover, our scheme can be used on a system interconnected by a k-stage network, $k > n$, as long as the routing scheme on this network is known. The principle of the reconfiguration of a system is conceptually to eliminate faulty components and, if necessary, sacrifice some usable components implicitly without knowing the actual locations of these components. A deadlock-free environment is provided for the reconfigured system such that the performance of the system is gracefully degraded. Deadlock-free shortest-path routing tables are obtained for processors in the surviving system to avoid possible deadlock traps which may be caused by the unidirectional communication rather than bidirectional communication between some processors. Because of the bit-oriented routing property of the Omega network [Law75], by generating the destination tag and referring to the routing table, a data packet from a source processor can always be routed through a proper intermediate processor during each pass through the faulty Omega network. Since the routing table provides information of multiple communication paths, a load-balancing scheme may also be employed to reduce traffic contention.

REFERENCES

[AgLe85] D.P. Agrawal and J.S. Leu, "Dynamic accessibility testing and path length optimization of multistage interconnection networks," *IEEE Trans. Comput.*, Vol. C-34, pp. 255-266, Mar. 1985.

[AgSw88] D.P. Agrawal and N.K. Swain, "Analysis and Design of Nonequivalent Multistage Interconnection Networks," *IEEE Trans. Comput.*, Vol. C-37, pp. 232-237, Feb. 1988.

[Agr80] D.P. Agrawal, "Automated testing of computer networks," *Proc. 1980 Int. Conf. Circ. Comput.*, pp. 717-720, Oct. 1980.

[Agr82] D.P. Agrawal, "Testing and fault-tolerance of multistage interconnection networks," *IEEE Computer,* Vol. 15, pp. 41-53, Apr. 1982.

[Agr83] D.P. Agrawal, "Graph theoretical analysis and design of multistage interconnection networks," *IEEE Trans. Comput.*, Vol. C-32, pp. 637-648, Jul. 1983.

[Bat76] K.E. Batcher, "The Flip Network in Startan," *Proc. Int. Conf. Parallel Processing*, pp. 65-71, Aug. 1976.

[BeFo88] J.C. Bermond and J.M. Fourneau, "Independent Connections: An Easy Characterization of Baseline-Equivalent Multistage Interconnection Networks," *Proc. of International conf. on Parallel Processing*, 1988, pp. 187-190.

[Bhu87] L.N. Bhuyan, "Interconnection networks for parallel and distributed processing," *IEEE Computer*, 20, pp. 9-12, Jun. 1987.

[DaBh85] C.R. Das and L.N. Bhuyan, "Reliability simulation of multiprocessor systems," *Proc. Int. Conf. Parallel Processing,* Aug. 1985, pp. 764-771.

[FaPr81] K.M. Falavarianai and D.K. Pradhan, "Fault-diagnosis of parallel processor interconnection networks," *Proc. 11th Annu. Int. Symp. Fault-Tolerant Comput.*, Jun. 1981.

[Fen74] T.Y. Feng, "Data Manipulating Functions in Parallel Processor and Their Implementations," *IEEE Trans. Comput.*, Vol. C-23, pp. 309-318, Mar. 1974.

[Fen81] T.Y. Feng, "A Survey of Interconnection Networks," *IEEE Computer*, Vol. 14, No. 12, pp. 12-27, Dec. 1981.

[FuAb83] W.K. Fuchs, J.A. Abraham and K.H. Huang, "Concurrent error detection in VLSI interconnection networks," *Proc. 1983 Int. Symp. Computer Architecture.*, pp. 309-315, 1983.

[GaMa88] I. Gazit and M. Malek, "Fault tolerance capabilities in multistage network-based multicomputer systems," *IEEE Trans. Comput*, Vol. 37, no. 7, pp 788-798, Jul. 1988.

[GoLi78] L.R. Goke and G.J. Lipovski, "Banyan Networks for Partitioning Multiprocessor Systems," *5th Annual Symposium on Computer Architecture*, pp. 21-28, Dec. 1978.

[HoSa78] E. Horowitz and S. Sahni, "Fundamentals of computer algorithms," *Computer Science Press,* 1978.

[HuTr86] S.T. Huang and S.K. Tripathi, "Finite State Model and Compatibility Theory: New Analysis Tools for Permutation Networks," *IEEE Trans. Comput.*, Vol.

C-35, pp. 591-601, Jul. 1986.

[Hwa84] K. Hwang and F.A. Briggs, "Computer Architecture and Parallel Processing," McGraw-Hill pub. 1984.

[KrSn86] C.P. Kruskal and M. Snir, "A Unified Theory of Interconnection Network Structure," *Th. Comp. Sci.*, Vol. 48, No. 1, pp. 75-94, 1986.

[Law75] D.H. Lawrie, "Access and alignment of data in an array processor," *IEEE Trans. Comput.*, Vol. C-24, pp. 1145-1155, Dec. 1975.

[Lee85] K. Y. Lee "On the rearrangeability of $2(\log_2 N)$ - 1 stage permutation networks,"*IEEE Trans. Comput.*, Vol. C-34, pp. 412-425, May 1985.

[Len78] J. Lenfant, "Parallel permutations of data: A Benes network control algorithm for frequently used permutations," *IEEE Trans. Comput.*, Vol. C-27, pp. 637-647, July 1978.

[NaSa81] D. Nassimi and S. Sahni, "A self-routing Benes network and parallel permutation algorithms," *IEEE Trans. Comput.*, Vol. C-30, pp. 241-249, May 1981.

[NaSo80] J.J. Narraway and K.M. So, "Fault diagnosis in inter-processor switching networks," *Proc. 1980 Int. Conf. Circ. Comput.*, pp. 750-753, Oct. 1980.

[Par80] D. S. Parker, "Notes on shuffle/exchange-type switching networks," *IEEE Trans. Comput.*, Vol. C-29, pp. 213-222, Mar. 1980.

[Pat81] J.H. Patel, "Processor-Memory Interconnections for Multiprocessors," *IEEE Trans. Comput.*, Vol. C-30, pp. 771-780, Oct. 1981.

[Pea77] M.C. Pease, "the Indirect binary n-cube Microprocessor Array," *IEEE Trans. Comput.*, Vol. C-26, pp. 458-473, May 1977.

[PrYe73] I.P. Preparata and R.T. Yeh, "Introduction to discrete structures for computer science and engineering," *Addison-Wesley Pub.* 1973.

[ReNi77] E. M. Reingold, J. Nievergelt, and N. Deo, *Combinatorial Algorithms: Theory and Practice*, Prentice-Hall Inc., Englewood Cliffs, New Jersey, 1977.

[ShHa84] J.P. Shen and J.P. Hayes, "Fault-tolerance of dynamic-full-access interconnection networks," *IEEE Trans. Comput.,* Vol. C-33, no. 3, pp. 241-248, Mar. 1984.

[Sie79] H.J. Siegel, "Interconnection Networks for SIMD Machines," *IEEE Computer*, Vol. 12, pp. 57-65, Jun. 1979.

[SiSm78] M.C. Pease, "the Indirect binary n-cube Microprocessor Array," *IEEE Trans. Comput.*, Vol. C-26, pp. 458-473, May 1977.

[Ste83] D. Steinberg, "Invariant properties of the shuffle-exchange and a simplified cost-effective version of the Omega network," *IEEE Trans. Comput.*, Vol. C-32, pp. 444-450, May 1983.

[Sto71] H. S. Stone, "Parallel processing with perfect shuffle," *IEEE Trans. Comput.*, Vol. C-20, pp. 153-161, Feb. 1971.

[TaNe83] S. Thanawastien and V.P. Nelson, "Optimal fault detection test sequences for shuffle/exchange networks," *Proc. 13th Annu. Int. Symp. Fault-Tolerant Comput.*, pp. 442-445, Jun. 1983.

[VaRa89] A. Varma and C.S. Raghavendra, "Fault-tolerant routing in multistage interconnection networks," *IEEE Trans. Comput.*, Vol. 38, pp. 385-393, Mar. 1989.

[WiCh78] D.A. Wismer and R. Chattergy, "Introduction to nonlinear optimization," *North-Holland Pub.* 1978.

[WuFe79] C.L. Wu and T.Y. Feng, "Fault-diagnosis for a class of multistage interconnection networks," *Proc. 1979 Int. Conf. Parallel Processing*, pp. 269-278, Aug. 1979.

[WuFe80] C.L. Wu and T.Y. Feng, "On a Class of Multistage Interconnection Networks," *IEEE Trans. Comput.*, Vol. C-29, pp. 694-702, Aug. 1980.

[WuFe81] C. Wu and T. Feng, "The universality of the shuffle-exchange network." *IEEE Trans. Comput.*. Vol. C-30, pp. 324-332, May 1981.

DISTRIBUTION LIST

addresses	number of copies
RADC/COAC ATTN: Jon Valente Griffiss AFB NY 13441-5700	30
Syracuse University Office of Sponsored Programs Skytop Office Building Skytop Road Syracuse NY 13210	5
RADC/DOVL Technical Library Griffiss AFB NY 13441-5700	1
Administrator Defense Technical Info Center DTIC-FDAC Cameron Station Building 5 Alexandria VA 22304-6145	2
Strategic Defense Initiative Office Office of the Secretary of Defense Wash DC 20301-7100	2
RADC/COAC Griffiss AFB NY 13441-5700	1
HQ USAF/SCTT Washington DC 20330-5190	1
SAF/AQSC Pentagon Rm 4D 269 Wash DC 20330	1

Naval Warfare Assessment Center GIDEP Operations Center/Code 30G ATTN: E Richards Corona CA 91720	1
HQ AFSC/XTH Andrews AFB MD 20334-5000	1
HQ SAC/SCPT OFFUTT AFB NE 68046	2
HQ TAC/DRIY ATTN: Maj. Divine Langley AFB VA 23665-5575	1
HQ TAC/DOA Langley AFB VA 23665-5554	1
ASD/ENEMS Wright-Patterson AFB OH 45433-6503	1
SM-ALC/MACEA ATTN: Danny McClure Bldg 237, MASOF McClellan AFB CA 95652	1
WRDC/AAAI-4 Wright-Patterson AFB OH 45433-6543	1

WRDC/AAAI-2 ATTN: Mr Franklin Hutson WPAFB OH 45433-6543	1
AFIT/LDEE Building 642, Area B Wright-Patterson AFB OH 45433-6583	1
WRDC/MLPO ATTN: D.L. Denison WPAFB OH 45433-6533	1
WRDC/MTEL Wright-Patterson AFB OH 45433	1
AAMRL/HE Wright-Patterson AFB OH 45433-6573	1
Air Force Human Resources Lab Technical Documents Center AFHRL/LRS-TDC Wright-Patterson AFB OH 45433	1
AUL/LSE Bldg 1405 Maxwell AFB AL 36112-5564	1
HQ ATC/TTOI ATTN: Lt Col Killian Randolph AFB TX 78150-5001	1

AFLMC/LGY 1
ATTN: Maj. Shaffer
Building 205
Gunter AFS AL 36114-6693

US Army Strategic Def 1
CSSD-IM-PA
PO Box 1500
Huntsville AL 35807-3801

Ofc of the Chief of Naval Operation 1
ATTN: William J.Cook
Navy Electromagnetic Spectrum Mgt
Room 5A678, Pentagon (OP-941)
Wash DC 20350

Commanding Officer 1
Naval Avionics Center
Library D/765
Indianapolis IN 46219-2139

Commanding Officer 1
Naval Ocean Systems Center
Technical Library
Code 96423
San Diego CA 92152-5000

Cmdr 1
Naval Weapons Center
Technical Library/C3431
China Lake CA 93555-6001

Superintendent 1
Code 1424
Naval Postgraduate School
Monterey CA 93943-5000

Space & Naval Warfare Systems Comm 1
Washington DC 20363-5100

CDR, U.S. Army Missile Command 2
Redstone Scientific Info Center
AMSMI-RD-CS-R/ILL Documents
Redstone Arsenal AL 35898-5241

Advisory Group on Electron Devices 2
201 Varick Street, Rm 1140
New York NY 10014

Los Alamos National Laboratory 1
Report Library
MS 5000
Los Alamos NM 87544

AEDC Library 1
Tech Files/MS-100
Arnold AFB TN 37389

Commander, USAG 1
ASQH-PCA-CRL/Tech Lib
Bldg 61801
Ft Huachuca AZ 85613-6000

1839 EIG/EIT 1
Keesler AFB MS 39534-6348

AFEWC/ESRI 3
San Antonio TX 78243-5000

ESD/XRR 1
Hanscom AFB MA 01731-5000

SEI JPO 1
ATTN: Major Charles J. Ryan
Carnegie Mellon University
Pittsburgh PA 15213-3890

Director NSA/CSS 1
T513/TDL
ATTN: D W Marjarum
Fort Meade MD 20755-6000

Director NSA/CSS 1
W157
9800 Savage Road
Fort Meade MD 21055-6000

NSA 1
ATTN: D. Alley
Div X911
9800 Savage Road
Ft Meade MD 20755-6000

Director 1
NSA/CSS
W11 DEFSMAC
ATTN: Mr. Mark E. Clesh
Fort George G. Meade MD 20755-6000

Director 1
NSA/CSS R12
ATTN: Mr. Dennis Heinbuch
9800 Savage Road
Fort George G. Meade MD 20755-6000

DoD 1
R31
9800 Savage Road
Ft. Meade MD 20755-6000

DIRNSA 1
R509
9800 Savage Road
Ft Meade MD 20775

Director 1
NSA/CSS
R08
Fort George G. Meade MD 20755-6000

DOD Computer Center 1
C/TIC
9800 Savage Road
Fort George G. Meade MD 20755-6000

SDI/S-P1-BM 1
ATTN: Cmdr Korajo
The Pentagon
Wash DC 20311-7100

SDIO/S-PL-BM 1
ATTN: Capt Johnson
The Pentagon
Wash DC 20301-7000

SDIO/S-PL-BM 1
ATTN: Lt Col Rindt
The Pentagon
Wash DC 20301-7100

IDA (SDIO Library) 1
ATTN: Mr. Albert Perrella
1801 N. Beauregard Street
Alexandria VA 22311

SAF/AQSD 1
ATTN: Maj M. K. Jones
The Pentagon
Wash DC 20330

AFSC/CV-D 1
ATTN: Lt Col Flynn
Andrews AFB MD 20334-5000

HQ SD/XP 1
ATTN: Col Heimach
PO Box 92960
Worldway Postal Center
Los Angeles CA 90009-2960

HQ SSD/CNC ATTN: Col O'Brien PO Box 92960 Worldway Postal Center Los Angeles CA 90009-2960	1
HQ SD/CNCI ATTN; Col Collins PO Box 92960 Worldway Postal Center Los Angeles CA 90009-2960	1
HQ SD/CNCIS ATTN; Lt Col Pennell PO Box 92960 Worldway Postal Center Los Angeles CA 90009-2960	1
ESD/AT ATTN: Col Ryan Hanscom AFB MA 01731-5000	1
ESD/ATS ATTN: Lt Col Oldenberg Hanscom AFB MA 01731-5000	1
ESD/ATN ATTN: Col Leib Hanscom AFB MA 01731-5000	1
AFSTC/XPX (Lt Col Detucci) Kirtland AFB NM 87117	1
AFSPACECOM/XPD ATTN: Maj Roger Hunter Peterson AFB CO 80914	1
GE SDI-SEI ATTN: Mr. Ron Harking 1737 Century Park West Bluebell PA 194422	1

MITRE Corp ATTN: Dr. Donna Cuomo Bedford MA 01730	1
SSD/CNI ATTN: Lt Col Joe Rouge P. O. Box 92960 Los Angeles AFB CA 90009-2960	1
NTB JPO ATTN: Maj Don Ravenscroft Falcon AFB CO 80912	1
Ford Aerospace Corp c/o Rockwell International ATTN: Dr. Joan Schulz 1250 Academy Park Loop Colorado Springs CO 80910	1
Essex Corp ATTN: Dr. Bob Mackie Human Factors Research Div 5775 Dawson Ave Goleta CA 93117	1
Naval Air Development Ctr ATTN: Dr. Mort Metersky Code 30D Warminster PA 189974	1
RJO Enterprises ATTN: Mr. Dave Israel 1225 Jefferson Davis HY Suite 300 Arlington VA 22202	1
GE SDI SEI ATTN: Mr. Bill Bensch 1707 Century Park West Bluebell PA 19422	1
HQ AFOTEC/OAHS ATTN: Dr. Samuel Charlton Kirtland AFB NM 87117	[illegible]

ESD/XTS 1
ATTN: Lt Col Joseph Toole
Hanscom AFB MA 01731

SDIO/ENA 1
ATTN: Col R. Worrell
Pentagon
Wash DC 20301

USA-SDC CSSD-H-SBE 1
ATTN: Mr. Doyle Thomas
Huntsville AL 35807

HQ AFSPACECOM/DOXP 1
ATTN; Capt Mark Terrace
Stop 7
Peterson AFB CO 80914

BBN Systems & Technology 1
ATTN: Dr. Dick Pew
70 Fawcett St
Cambridge MA 02138

ESD/XTI 1
ATTN: Lt Col Paul Monico
Hanscom AFB MA 01730

CSSD-H-S3 1
ATTN: Mr. Larry Tubbs
Commander USA SDC
PO Box 1500
Huntsville AL 35807

USSPACECOM/J5B 1
ATTN: Lt Col Harold Stanley
Peterson AFB CO 80914

NTB JPO 1
ATTN: Mr. Nat Sojouner
Falcon AFB CO 80912